Distributed Constrained Optimization with Semicoordinate Transformations

William Macready (wgm@email.arc.nasa.gov)
David Wolpert (dhw@email.arc.nasa.gov)

NASA Ames Research Center,
MailStop 269-4,
Moffett Field, CA, 94035

March 1, 2005

Abstract

Recent work has shown how information theory extends conventional full-rationality game theory to allow bounded rational agents. The associated mathematical framework can be used to solve constrained optimization problems. This is done by translating the problem into an iterated game, where each agent controls a different variable of the problem, so that the joint probability distribution across the agents' moves gives an expected value of the objective function. The dynamics of the agents is designed to minimize a Lagrangian function of that joint distribution. Here we illustrate how the updating of the Lagrange parameters in the Lagrangian is a form of automated annealing, which focuses the joint distribution more and more tightly about the joint moves that optimize the objective function. We then investigate the use of "semicoordinate" variable transformations. These separate the joint state of the agents from the variables of the optimization problem, with the two connected by an onto mapping. We present experiments illustrating the ability of such transformations to facilitate optimization. We focus on the special kind of transformation in which the statistically independent states of the agents induces a mixture distribution over the optimization variables. Computer experiment illustrate this for k-sat constraint satisfaction problems and for unconstrained minimization of NK functions.

Subject Classification: programming: nonlinear, algorithms, theory; probability: applications

Area of Review: optimization

1 Introduction

1.1 Distributed optimization and control with Probability Collectives

As first described in (Wolpert 2003*a*, Wolpert 2004*a*), it turns out that one can translate many of the concepts from statistical physics, game theory, distributed optimization and distributed control into one another. This translation is based on the fact that those concepts all involve distributed systems in which the random variables are, at any single instant, statistically independent. (What is coupled is instead the distributions of those variables.) Using this translation, one can transfer theory and techniques between those fields, creating a large common mathematics that connects them. This common mathematics is known as Probability Collectives (PC). Its unifying concern is the set of probability distributions that govern any particular distributed system, and how to manipulate those distributions to optimize one or more objective functions. See (Wolpert, Tumer & Bandari 2003, Wolpert & Tumer 2001) for earlier, less formal work on this topic.

In this paper we consider the use of PC to solve constrained optimization and/or control problems. Reflecting the focus of PC on distributed systems, its use for such problems is particularly appropriate when the variables in the collective are spread across many physically separated agents with limited inter-agent communication (e.g., in a distributed design or supply chain application, or distributed control). A general advantage of PC for such problems is that since they work with probabilities rather than the underlying variables, they can be implemented for arbitrary types of the underlying variables. This same characteristic also means they provides multiple solutions, each of which is robust, along with sensitivity information concerning those solutions. An advantage particulary relevant to optimization is that the distributed PC algorithm can often be implemented on a parallel computer. An advantage particularly relevant to control problems is that PC algorithms can, if desired, be used without any modelling assumptions about the (stochastic) system being controlled. These advantages are discussed in more detail below.

1.2 The Probability Collectives Approach

Broadly speaking, the PC approach to optimization/control is as follows. First one maps the provided problem into a multi-agent collective. In the simplest version of this process one assigns a separate agent of the collective to determine the value of each of the variables $x_i \in \mathcal{X}_i$ in the

problem that we control. So for example if the i'th variable can only take on a finite number of values, those $|\mathcal{X}_i|$ possible values constitute the possible moves of the i'th agent.[1] The value of the joint set of n variables (agents) describing the system is then $\mathbf{x} = [x_1, \cdots, x_n] \in \mathcal{X}$ with $\mathcal{X} \triangleq \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$.[2]

Unlike many optimization methods, in PC the variables are not manipulated directly. Rather a probability distribution is what is manipulated. To avoid combinatorial explosions as the number of dimensions of $\mathcal{X}$ grows, we must restrict attention to a low-dimensional subset of the space of all probability distributions. We indicate this by writing our distributions as $q \in \mathcal{Q}$ over $\mathcal{X}$. The manipulation of that q proceeds through an iterative process. The ultimate goal of this process is to induce a distribution that is highly peaked about the $\mathbf{x}$ optimizing the objective function $G(\mathbf{x})$, sometimes called the *world cost* or *world utility* function. (In this paper we only consider problems with a single overall objective function, and we arbitrarily choose lower values to be better, even when using the term "utility".)

In the precise algorithms investigated here, at the start of any iteration a single Lagrangian function of q, $\mathcal{L} : \mathcal{Q} \rightarrow \mathbb{R}$, is specified, based on $G(\mathbf{x})$ and the associated constraints of the optimization problem. Rather than minimize the objective function over the space $\mathcal{X}$, the algorithm minimizes that Lagrangian over $q \in \mathcal{Q}$. This is done by direct manipulation of the components of q by the agents.

After such a minimization of a Lagrangian, one modifies the Lagrangian slightly. This is done so that the q optimizing the new Lagrangian is more tightly concentrated about $\mathbf{x}$ that solve our optimization problem than is the current q. One then uses the current q as the starting point for another process of having the agents minimize a Lagrangian, this time having them work on that new Lagrangian.

At the end of a sequence of such iterations one ends up with a final q. That q is then used to determine a final answer in $\mathcal{X}$, e.g., by sampling q, evaluating its mode, evaluating its mean (if that is defined), etc. For a properly chosen sequence of Lagrangians and algorithm for minimizing the Lagrangians, this last step should, with high probability, provide the desired optimal point in $\mathcal{X}$.

For the class of Lagrangians used in this paper, the sequence of minimizations of Lagrangians is closely related to simulated annealing. The difference is that in simulated annealing an inefficient Metropolis sampling process is used to implicitly descend each iteration's Lagrangian. By explicitly

manipulating q, PC allows for more efficient descent.

In this paper we shall consider the case where $\mathcal{Q}$ is a product space, $q(\mathbf{x}) = \prod_i q_i(x_i)$. The associated formulation of PC is sometimes called "Product Distribution" theory. It corresponds to noncooperative game theory, with each q_i being agent i's "mixed strategy" (Wolpert 2004*a*, Fudenberg & Tirole 1991). Our particular focus is the use of such product distributions when $\mathcal{X}$ is not the same as the ultimate space of the optimization variables, $\mathcal{Z}$. In this formulation — a modification of what was presented above — there is an intermediate mapping from $\mathcal{X} \rightarrow \mathcal{Z}$, and the provided G is actually a function over $\mathcal{Z}$, not (directly) over $\mathcal{X}$. Such intermediate mappings are called semicoordinate systems, and going from one to another is a semicoordinate transformation. As elaborated below, such transformations allow arbitrary coupling among the variables in $\mathcal{Z}$ while preserving many of the computational advantages of using product distributions over $\mathcal{X}$.

1.3 Advantages of Probability Collectives

There are many advantages to working with distribution in $\mathcal{Q}$ rather than points in $\mathcal{X}$. Usually the support of q is all of $\mathcal{X}$, i.e., the q minimizing the Lagrangian lies in the interior of the unit simplices giving $\mathcal{Q}$. Conversely, any element of $\mathcal{X}$ can be viewed as a probability distribution on the edge (a vertex) of those simplices. So working with $\mathcal{X}$ is a special case of working with $\mathcal{Q}$, where one sticks to the vertices of $\mathcal{Q}$. In this, optimizing over $\mathcal{Q}$ rather than $\mathcal{X}$ is analogous to interior point methods. Due to the breadth of the support of q, minimizing over it can also be viewed as a way to allow information from the value of the objective function at all $\mathbf{x} \in \mathcal{X}$ to be exploited simultaneously.

Another advantage, alluded to above, is that by working with distributions $\mathcal{Q}$ rather than the space $\mathcal{X}$, the same general PC approach can be used for essentially any $\mathcal{X}$, be it continuous, discrete, time-extended, mixtures of these, etc. (Formally, those different spaces just correspond to different probability measures, as far as PC is concerned.) For expository simplicity though, here we will work with finite $\mathcal{X}$, and therefore have probability distributions rather than density functions, sums rather than integrals, etc. See in particular (Bieniawski & Wolpert 2004*a*, Wolpert & Bieniawski 2004*a*, Wolpert 2004*b*, Wolpert 2004*c*) for analysis explicitly for the case of infinite $\mathcal{X}$.

Yet another advantage arises from the fact that even when $\mathcal{X}$ is finite, $q \in \mathcal{Q}$ is a vector in

a Euclidean space. Accordingly the Lagrangian we are minimizing is a real-valued function of a Euclidean vector. This means PC allows us to leverage the power of descent schemes for continuous spaces like gradient descent or Newton's method — even if $\mathcal{X}$ is a categorical, finite space. So with PC, schemes like "gradient descent for categorical variables" are perfectly well-defined.

While the Lagrangians can be based on prior knowledge or modelling assumptions concerning the problem, they need not be. Nor does optimization of a Lagrangian require control of all variables $\mathcal{X}$ (i.e., some of the variables can be noisy). This allows PC to be very broadly applicable.

1.4 Connection with other sciences

A more general advantage of PC is how it relates seemingly disparate disciplines to one another. In particular, it can be motivated by using information theory to relate bounded rational game theory to statistical physics (Wolpert 2003*a*, Wolpert 2004*a*). This allows techniques from one field to be imported into the other field. For example, as illustrated below, the grand canonical ensemble of physics can be imported into noncooperative game theory to analyze games having stochastic numbers of the players of various types.

To review, a noncooperative game consists of a sequence of stages. At the beginning of each stage every agent (aka "player") sets a probability distribution (its "mixed strategy") over its moves (Fudenberg & Tirole 1991, Aumann & Hart 1992, Basar & Olsder 1999, Fudenberg & Levine 1998). The joint move at the stage is then formed by agents simultaneously sampling their mixed strategies at that stage. So the moves those agents make at any particular stage of the game are statistically independent and the distribution of the joint-moves at any stage is a product distribution — just like in PD theory.

This does not mean that the moves of the agents across all time are statistically independent however. At each stage of the game each agent will set its mixed strategy based on information gleaned from preceding stages, information that in general will reflect the earlier moves of the other agents. So the agents are coupled indirectly, across time, via the updating of the $\{q_i\}_{i=1}^n$ at the end of each stage.

Analogously, consider again the iterative PD algorithm outlined above, and in particular the process of optimizing the Lagrangian within some particular single iteration. Typically that process proceeds by successively modifying q across a sequence of timesteps. In each of those timesteps

$q(\mathbf{x}) = \prod_i q_i(x_i)$ is first sampled, and then it is updated based on all previous samples. So just like in a noncooperative game there is no direct coupling of the values of the underlying variables $\{x_i\}$at any particular timestep (q is a product distribution). Rather just like in a noncooperative game, the variables are indirectly coupled, across time (i.e., across timesteps of the optimization), via coupling of the distributions $q_i(x_i)$ at different timesteps.

In addition, information theory can be used to show that the bounded rational equilibrium of any noncooperative game is the q optimizing an associated "maxent Lagrangian" $\mathcal{L}(q)$ (Wolpert 2004a). (That Lagrangian is minimized by the distribution that has maximal entropy while being consistent with specified values of the average payoffs of the agents.) This Lagrangian turns out to be exactly the one that arises in the version of PC considered in this paper. So bounded rational game theory is an instance of PC.

Now in statistical physics often one wishes to find the distribution out of an allowed set of distributions (e.g., $\mathcal{Q}$) with minimal distance to a fixed target distribution $p \in \mathcal{P}$, the space of all possible distributions over $\mathcal{X}$. Perhaps the most popular choice for a distance measure between distributions is the Kullback-Leibler (KL) distance[3]: $D(q\|p) \triangleq \sum_{\mathbf{x}} q(\mathbf{x}) \ln\big(q(\mathbf{x})/p(\mathbf{x})\big)$ (Cover & Thomas 1991). As the KL distance is not symmetric in its arguments p and q we shall refer to $D(q\|p)$ as the qp KL distance (this is also sometimes called the exclusive KL distance), and $D(p\|q)$ as the pq distance (also sometimes called the inclusive KL distance).

Typically in physics p is given by one of the statistical "ensembles". An important example of such KL minimization arises with the Boltzmann distribution of the canonical ensemble: $p(\mathbf{x}) \propto \exp[-H(\mathbf{x})/T]$, where H is the "Hamiltonian" of the system. The KL distance $D(q||p)$ to the Boltzmann distribution is proportional to the Gibbs free energy of statistical physics. This free energy turns out to be identical to the maxent Lagrangian considered in this paper. Stated differently, if one solves for the distribution q from one's set that minimizes qp KL distance to the Boltzmann distribution, one gets the distribution from one's set having maximal entropy, subject to the constraint of having a specified expected value of H. When the set of distributions one's considering is $\mathcal{Q}$, the set of product distributions, this q minimizing qp KL distance to p is called a "mean-field approximation" to p. So mean-field theory is an instance of PC.

This illustrates that bounded rational games and the mean-field approximation to Boltzmann distributions are essentially identical. To relate them one equates H with a common payoff function

G. The equivalence is completed by then identifying each (independent) agent with a different one of the (independent) physical variables in the argument of the Hamiltonian.[4]

This connection between these fields allows us to exploit techniques from statistical physics in bounded rational game theory. For example, as mentioned above, rather than the canonical ensemble, we can apply the grand canonical ensemble to bounded rational games. This allows us to consider games in which the number of players of each type is stochastic (Wolpert 2004*a*).

1.5 The contribution of this paper

Use of a product distribution space $\mathcal{Q}$ for optimization/control is consistent with game theory (and more generally multi-agent systems). This choice also results in a highly parallel algorithm, and is well-suited to control problems that are inherently distributed. Nonetheless, other concerns may dictate different $\mathcal{Q}$. In particular, in many optimization tasks we seek multiple solutions far apart from one another. For example, in Constraint Satisfaction Problems (CSPs) (Dechter 2003), the goal is to identify all feasible solutions which satisfy a set of constraints, or to show that none exist. Typically when there are multiple feasible solutions they are vary far from one another. For small problem instances exhaustive enumeration techniques like branch-and-bound are typically used to identify all such feasible solutions; we are interested in larger problems.

In cases like these, where we desire multiple far-apart solutions, use of PC with a product distribution may be a poor choice. The problem is that if each distribution q_i is peaked about every value of x_i which occurs in at least one of the multiple solutions, then in general there will be spurious peaks in the product $q(\mathbf{x}) = \prod q_i(x_i)$, i.e., $q(\mathbf{x})$ may be peaked about some $\mathbf{x}$ that are not solutions. On the other hand the alternative scenario, where each q_i is only peaked about a few of the solutions, does not provide us the desired many solutions. To address this we might descend the Lagrangian many times, beginning from different starting points (i.e., different initial q). However there is no guarantee that multiple runs will each generate different solutions.

PC offers a simple solution to this problem that allows one to still use product distributions: extend the event space underlying our product distributions so that a single game provides multiple distinct solutions to our optimization problem at once. Formally, this is a semicoordinate transformation. Intuitively speaking, the transformation considered here recasts the problem in terms of a "meta-game" by cloning the original game into several simultaneous games, with an

independent set of agents for each game. We also have a supervisor agent who chooses what game is to be played. We then form a Lagrangian for the meta-game that is biased towards having any agents that control the same variable in different games have different mixed strategies from one another. The joint strategies for each of the separate games in the meta-game then give us our set of multiple solutions to the original game. The supervisor agent sets the probability distribution of which such solution is used. Since in general the resultant distribution across the variables being optimized (i.e., across $\mathcal{Z}$) cannot be written as a single product distribution, it provides coupling among those variables.

More precisely, recall that the space of arguments to our objective function is $\mathcal{Z}$, and our product distributions are instead over $\mathcal{X}$, with the "semicoordinate system" being the map from this space to $\mathcal{Z}$ (Wolpert & Bieniawski 2004*a*, Wolpert 2004*d*). Before transformation of the semicoordinate system, $\mathcal{X} = \mathcal{Z}$, and product distributions over $\mathcal{X}$ cannot give coupled distributions over $\mathcal{Z}$. However we will change $\mathcal{X}$ from $\mathcal{Z}$ and change the semicoordinate system in an associated way. We then consider product distributions over the new $\mathcal{X}$ (i.e., the noncooperative game is played in $\mathcal{X}$, not $\mathcal{Z}$). By appropriate choice of the semicoordinate transformation, such distributions corresponds to coupled distributions across $\mathcal{Z}$. In general any Bayes net topology can be achieved with an appropriate semicoordinate transformation (Wolpert 2004*d*, Wolpert & Bieniawski 2004*a*). Different product distributions over $\mathcal{Z}$ correspond to different Bayes nets having that same topology.

Here we consider a $\mathcal{X}$ that results in a mixture of M product distributions $\mathcal{Z}$, (Macready & Wolpert 2004*b*)

$$q(\mathbf{z}) = \sum_{m=1}^{M} q^0(m) q^m(\mathbf{z}).$$

Intuitively, q^0 is the distribution over the moves of the supervisor agent, with m being the game that agent chooses. This allows for the determination of M solutions at once. At the same time, due to the entropy term in the Lagrangian, it "pushes" the separate products $q^m(\mathbf{z})$ in the mixture apart. This biases the algorithm to trying to find separated solutions, as desired.

In Sec. 2 we review how one arrives at the Lagrangian considered in this paper, the maxent Lagrangian. Then in Sec. 3 we review two elementary techniques introduced in (Wolpert & Bieniawski 2004*b*, Wolpert 2003*a*, Wolpert 2004*b*) for updating a product distribution q to minimize the associated Lagrangian. In the experiments reported below, all terms in those update rules can

be calculated in closed form. This is not true in general though. In Sec. 4 we review a set of Monte-Carlo techniques for addressing such general scenarios.

We review semicoordinate transformations in Sec. 5, with particular attention for how mixture models may be seen as a product distributions over a different space. In Sec.6 we analyze the minimization of the maxent Lagrangian associated with mixture-inducing semicoordinate transformations. In that section we also relate our maxent Lagrangian for mixture distributions to the Jensen-Shannon distance over $\mathcal{X}$. Experimental validation of these techniques is then presented for the k-satisfiability CSP problem (section 7.1) and the NK (section 7.2) optimization problems. These sections consider both situations where the semicoordinate transformation is fixed upfront and those where it is found dynamically.

We end with a synopsis of some other techniques for updating a product distribution q to minimize the associated Lagrangian. This synopsis serves as the basis for a discussion of the relationship between PC and other techniques. The distinguishing feature of PC is that it does not treat the variable $\mathbf{x}$ as the fundamental object to be optimized, but rather the distribution across it, q. Furthermore, samples of that distribution are only used if necessary to estimate quantities that cannot be evaluated other ways. The fundamental objective function is stated in terms of q.

It should be emphasized that like all of PC, the techniques presented in this paper can readily be applied to many problems other than constrained optimization. For example, PC provides a natural improvement to the Metropolis sampling algorithm (Wolpert & Lee 2004), which the techniques of this paper should be able to improve further. See (Antoine, Bieniawski, Kroo & Wolpert 2004, Wolpert & Bieniawski 2004*a*, Bieniawski, Wolpert & Kroo 2004, Bieniawski & Wolpert 2004*b*) for other examples and experiments.

2 The Lagrangian for Product Distributions

We begin by considering the case of the identity semicoordinate system, $\mathcal{X} = \mathcal{Z}$. As discussed above, we consider qp distance to the T-parameterized Boltzmann distribution $p(\mathbf{x}) = \exp[-G(\mathbf{x})/T]/Z(T)$ where $Z(T)$ is a normalization constant. At low T the Boltzmann distribution is concentrated on $\mathbf{x}$ having low G values, so that the product distribution with minimal qp distance to it would be expected to have the same behavior. Accordingly, one would expect that by taking qp KL distance

to this distribution as one's Lagrangian, and modifying the Lagrangian from one iteration to the next by lowering T, one should end up at a q concentrated on $\mathbf{x}$ having low G values. (See (Wolpert & Bieniawski 2004*b*, Wolpert 2004*b*, Wolpert 2004*c*) for a more detailed formal justification of using this Lagrangian based on solving constrained optimization problems with Lagrange parameters.)

More precisely, the qp KL distance to the Boltzmann distribution is the maxent Lagrangian,

$$\mathcal{L}(q) = \mathbb{E}_q(G) - TS(q) \tag{1}$$

up to irrelevant additive and multiplicative constants. Equivalently, we can write it as

$$\mathcal{L}(q) = \beta\mathbb{E}_q(G) - S(q) \tag{2}$$

where $\beta \triangleq 1/T$, up to an irrelevant overall constant. In these equations the inner product $\mathbb{E}_q(G) \triangleq \sum_{\mathbf{x}} q(\mathbf{x})G(\mathbf{x})$ is the expected value of G under q, and $S(q) \triangleq -\sum_{\mathbf{x}} q(\mathbf{x}) \ln q(\mathbf{x})$ is the Shannon entropy of q.

For q's which are product distributions $S(q) = \sum_i S(q_i)$ where $S(q_i) = -\sum_{x_i} q_i(x_i) \ln q_i(x_i)$. Accordingly, we can view the maxent Lagrangian as equivalent to a set of Lagrangians, $\mathcal{L}_i(q) = \sum_{x_i} \mathbb{E}_{q_{-i}}[G(x_i, \mathbf{x}_{-i})]q_i(x_i) - TS_i(q_i)$, one such Lagrangian for each agent i so that $\mathcal{L}(q) = \sum_{i=1}^{n} \mathcal{L}_i(q)$.[5] The first term in $\mathcal{L}$ is minimized by having perfectly rational players, i.e. by players who concentrate all their probability on the moves that are best for them, given the distributions over the agents. The second term is minimized by perfectly irrational players, i.e., by a perfectly uniform joint mixed strategy q. So T specifies the balance between the rational and irrational behavior of the players. In particular, for $T \to 0$, by minimizing the Lagrangian we recover the Nash equilibria of the game. Alternatively, from a statistical physics perspective, where T is the temperature of the system, this maxent Lagrangian is simply the Gibbs free energy for the Hamiltonian G.

Since we are interested in problems with constraints, we replace G in Eqs. (1) and (2) with

$$G(\mathbf{x}) + \sum_{a=1}^{C} \lambda_a c_a(\mathbf{x}) \tag{3}$$

where G is the original objective function and the c_a are the set of C equality constraint functions that are required to be equal to zero. The λ_a are the Lagrange multipliers that are used to enforce

the constraints. Collectively, we refer to the Lagrange multipliers with the C-vector $\boldsymbol{\lambda}$.

In CSP's we take the original objective function to be the constant function 0. In addition, the constraints are all equality constraints, so a saddle point of the Lagrangian over the space of possible q and $\boldsymbol{\lambda}$ is a solution of our problem. Note though that we don't have to find the exact saddle point; in general sampling from a q close to the saddle point will give us the $\mathbf{x}$'s we seek.

An alternative approach to incorporating constraints would start by weakening them so that they can be violated. We would then iteratively anneal down those weaknesses, i.e., strengthen the constraints, to where they are not violated. In this approach we replace the maxent Lagrangian formulation encapsulated in Eq.'s (2) and (3) with

$$\mathcal{L}(q, \beta, \boldsymbol{\lambda}) = \beta[\mathbb{E}_q(G) - \gamma_G] + \sum_a \lambda_a [\mathbb{E}_q(c_a) - \gamma_a] - S(q). \tag{4}$$

In each iteration of the algorithm β, $\boldsymbol{\lambda}$ are treated as Lagrange parameters and one solves for their values that enforce the equality constraints $\mathbb{E}_q(G) = \gamma_G$, and the C constraints $\mathbb{E}_q(c_a) = \gamma_a$ while also minimizing $\mathcal{L}(q, \beta, \boldsymbol{\lambda})$. In the usual way, since our constraints are all equalities, one can do this by finding saddle points of $\mathcal{L}(q, \beta, \boldsymbol{\lambda})$. The next iteration would then start by modifying our Lagrangian by shrinking the values γ_G, $\{\gamma_a\}$ slightly before proceeding to a new process of finding a saddle point.

For pedagogical simplicity, here we do not consider this alternative approach, but concentrate on the Lagrangian of Eq. (1) with the G of Eq. (3). Note that the vectors $\{q_i\}$ must be probability distributions. So there are implicit constraints our solution must satisfy: $0 \leq q_i(x_i) \leq 1$ for all i and x_i, and $\sum_{x_i} q_i(x_i) = 1$ for all i. To reduce the size of our equations we do not explicitly write those constraints.

3 Minimizing the maxent Lagrangian

For fixed β, our task is to find a saddle point of $\mathcal{L}(q, \boldsymbol{\lambda})$. In "first order methods" such a saddle point is found by iterating a two-step process. In the first step the Lagrange parameters $\boldsymbol{\lambda}$ are fixed and one solves for the q that minimizes the associated $\mathcal{L}$.[6] In the second step one then freezes that q and updates the Lagrange parameters. There are more sophisticated ways of finding saddle

points (Grantham 2004), and more generally one can use modified versions of the Lagrangian (e.g., an augmented Lagrangian (Bertsekas 1996)). Here for pedagogical simplicity we do not consider such more sophisticated approaches.

In this section we review two approaches to finding the $\{q_i\}$ for fixed Lagrange multipliers $\boldsymbol{\lambda}$. We also describe our approach for the second step of the first order method, i.e., we describe how we use gradient ascent to update the Lagrange multipliers λ_a for fixed q. See (Wolpert 2003*a*, Wolpert & Bieniawski 2004*b*, Wolpert 2004*b*) for further discussion of these approaches as well as the many others one can use.

3.1 Brouwer Updating

At each step t the direction in the simplex $\mathcal{Q}$ that, to first order, maximizes the drop in $\mathcal{L}$ is given by (-1 times)

$$\tilde{\nabla}_q \mathcal{L}(q) \triangleq \nabla_q \mathcal{L}(q) - \eta(q). \tag{5}$$

In this equation the $q_i(x_i)$ component of the gradient (one for every agent i and every possible move x_i by the agent) is

$$[\nabla_q \mathcal{L}(q)]_{q_i(x_i)} = \frac{\partial \mathcal{L}}{\partial q_i(x_i)} = \mathbb{E}_{q_{-i}}(G|x_i) + T \ln[q_i(x_i)] \tag{6}$$

where

$$\mathbb{E}_{q_{-i}}(G|x_i) = \sum_{\mathbf{x}_{-i}} q_{-i}(\mathbf{x}_{-i}) G(x_i, \mathbf{x}_{-i})$$

with $\mathbf{x}_{-i} \triangleq [x_1, \cdots, x_{i-1}, x_{i+1}, \cdots, x_n]$ and $q_{-i}(\mathbf{x}_{-i}) \triangleq \prod_{j=1|j\neq i}^{n} q_j(x_j)$. $\eta(q)$ is the vector that needs to be added to $\nabla_q \mathcal{L}(q)$ to get it back into $\mathcal{Q}$.[7] The $q_i(x_i)$ component of $\eta(q)$, is equal to

$$[\eta(q)]_{q_i(x_i)} = \eta_i(q) \triangleq \frac{1}{|\mathcal{X}_i|} \sum_{x_i'} [\nabla_q \mathcal{L}(q)]_{q_i(x_i')} \tag{7}$$

where $|\mathcal{X}_i|$ is the number of possible moves x_i. Not that for any agent i, all of the associated components of $\eta(q)$, namely $q_i(x_1), \cdots, q_i(x_{|\mathcal{X}_i|})$, share the same value $\eta_i(q)$. This choice ensures that $\sum_{x_i} q_i(x_i) = 1$ after the gradient update to the values $q_i(x_i)$.

The expression in Eq. (3.1) is the expected payoff to agent i when it plays move x_i, under the

distribution q_{-i} across the moves of all other agents. Setting $\tilde{\nabla}_q \mathcal{L}(q)$ to zero gives the solution

$$q_i^{t+1}(x_i) \propto \exp\left[-\mathbb{E}_{q_{-i}^t}(G|x_i)/T\right] \tag{8}$$

Brouwer's fixed point theorem guarantees the solution of Eq. (8) exists for any G (Wolpert 2004*a*, Wolpert 2003*a*). Hence we call update rules based on this equation *Brouwer updating*.

Brouwer updating can be done in parallel on all the agents. One problem that can arise here is "thrashing". Each agent i is adopting the q_i that would be optimal if all the other agents didn't change their distributions. However they do change their distributions, and thereby at least partially confound agent i. One way to address this problem is to have agent i not use the current value $\mathbb{E}_{q_{-i}^t}(G|x_i)$ alone to update $q_i^t(x_i)$, but rather use a weighted average of all values $\mathbb{E}_{q_{-i}^{t'}}(G|x_i)$ for $t' \leq t$, with the weights shrinking the further into the past one goes. This introduces an inertia effect which helps to stabilize the updating. (Indeed, in the continuum-time limit, this weighting becomes the replicator dynamics (Wolpert 2004*d*).)

A similar idea is to have agent i use the current $\mathbb{E}_{q_{-i}^t}(G|x_i)$ alone, but have it only move part of the way the parallel Brouwer update recommends. Whether one moves all the way or only part-way, what agent i is interested in is what distribution will be optimal *for the next distributions of the other agents*. Accordingly, it makes sense to have agent i predict, using standard time-series tools, what those future distributions will be. This amounts to predicting what the next vector of values of $\mathbb{E}_{q_{-i}^t}(G|x_i)$ will be, based on seeing how that vector has evolved in the recent past. See (Shamma & Arslan 2004) for related ideas.

Another way of circumventing thrashing is to have the agents update their distributions serially (one after the other) rather than in parallel. See (Wolpert & Bieniawski 2004*a*) for a description of various kinds of serial schemes, as well as a discussion of partial serial, partial parallel algorithms.

3.2 Nearest-Newton Updating

To evaluate the gradient one only needs to evaluate or estimate the terms $\mathbb{E}_{q_{-i}^t}(G|x_i)$ for all agents (see below and (Wolpert 2003*a*, Wolpert 2004*a*)). So gradient descent is typically straight-forward. It is also usually simple to evaluate the Hessian of our Lagrangian. However conventional Newton's descent is often intractable for large systems, since that Hessian is so big that inverting often it

isn't feasible.

Of course there are schemes like conjugate gradient or quasi-Newton that can exploit second order information even when the Hessian cannot be inverted. However the special structure of the Lagrangian also allows second order information to be used for a simple variant of Newton descent. The associated update rule is called *Nearest-Newton* updating (Wolpert & Bieniawski 2004*b*); we review it here.

To derive Nearest-Newton we begin by considering the Lagrangian $\mathbb{E}_\pi(G) - TS(\pi)$, for an *unrestricted* probability distribution π.[8] This Lagrangian is a convex function of π with a diagonal Hessian. So given a current distribution π^t we can make an unrestricted Newton step of this Lagrangian to a new distribution π^{t+1}. That new distribution typically is not in $\mathcal{Q}$, even if the starting distribution is. However we can solve for the $q^{t+1} \in \mathcal{Q}$ that is nearest to π^{t+1}, for example by finding the $q^{t+1} \in \mathcal{Q}$ that minimizes qp KL distance $D(p||q)$ to that new point.

More precisely, the Hessian of $\mathbb{E}_\pi(G) - TS(\pi)$, $\partial^2 \mathcal{L}/\partial\pi(\mathbf{x})\partial\pi(\mathbf{x}')$, is diagonal, and so is simply inverted. This gives the Newton update for π^t:

$$\pi^{t+1}(\mathbf{x}) = \pi^t(\mathbf{x}) - \alpha_q^t \pi^t(\mathbf{x}) \left[\frac{G(\mathbf{x}) - \mathbb{E}_{\pi^t}(G)}{T} + S(\pi^t) + \ln \pi^t(\mathbf{x}) \right]$$

which is normalized if π^t is normalized and where α_q^t is a step size. As π^t will typically not belong to $\mathcal{Q}$ we find the product distribution nearest to π^{t+1} by minimizing the KL distance $D(\pi^{t+1}\|q)$ with respect to q. The result is that $q_i(x_i) = \pi_i^{t+1}(x_i)$, i.e. q_i is the marginal of π^{t+1} given by integrating it over $\mathbf{x}_{-i}$.

Thus, whenever π^t itself is a product distribution, the update rule for $q_i(x_i)$ is

$$q_i^{t+1}(x_i) = q_i^t(x_i) - \alpha_q^t q_i^t(x_i) \left[\frac{\mathbb{E}_{q^t_{-i}}(G|x_i) - \mathbb{E}_{q^t}(G)}{T} + S(q_i) + \ln q_i^t(x_i) \right]. \tag{9}$$

This update maintains the normalization of q_i, but may make one or more $q_i^{t+1}(x_i)$ greater than 1 or less than 0. In such cases we set q_i^{t+1} to be valid product distribution nearest in Euclidean distance (rather than KL distance) to the suggested Newton update.

3.3 Updating Lagrange Multipliers

In order to satisfy the imposed optimization constraints $\{c_a(\mathbf{x})\}$ we must also update the Lagrange multipliers. To minimize communication between agents this is done in the simplest possible way – by gradient descent. Taking the partial derivatives with respect to λ_a gives the update rule

$$\lambda_a^{t+1} = \lambda_a^t + \alpha_\lambda^t \mathbb{E}_{q_*^t}(c_a(\mathbf{x})) \tag{10}$$

where α_λ^t is a step size and q_*^t is the local minimizer of $\mathcal{L}$ determined as above at the old settings, $\boldsymbol{\lambda}^t$, of the multipliers.

3.4 Other descent schemes

It should be emphasized that PC encompasses many approaches to optimization of the Lagrangian that differ from those used here. For example, in (Bieniawski & Wolpert 2004*a*, Wolpert 2004*d*) there is discussion of alternative types of descent algorithms that are related to block relaxation, as well as to the fictitious play algorithm of game theory (Fudenberg & Tirole 1991, Shamma & Arslan 2004) and multi-agent reinforcement learning algorithms like those in collective intelligence (Wolpert & Tumer 2001, Wolpert et al. 2003).

As another example, see (Wolpert & Bieniawski 2004*b*, Wolpert 2004*a*) for discussions of using pq KL distance (i.e., $D(p||q)$) rather than qp distance. Interestingly, as discussed below, that alternative distance must be used even for descent of qp distance, if one wishes to use 2nd order descent schemes. (Wolpert 2004*b*, Wolpert 2004*c*) discusses using non-Boltzmann target distributions p, and many other options for what functional(s) to descend.

4 Statistical estimation to update q

Using either of the update rules Eqs. (8) or (9) requires knowing $\mathbb{E}_{q_{-i}^t}(G|x_i)$, defined in Eq. (3.1). However, often we cannot efficiently calculate all the terms $\mathbb{E}_{q_{-i}^t}(G|x_i)$. To use our update rules in such situations we can use Monte Carlo sampling, as described in this section.

4.1 Monte Carlo sampling

In the Monte Carlo approach, at each timestep every agent i samples its distribution q_i to get a point x_i. Since we have a product distribution q, these samples provides us with a sample $\mathbf{x}$ of the full joint distribution q. By repeating this process L times we get a "block" of such joint samples. The G values in that block can be used by each agent separately to estimate its updating values $\mathbb{E}_{q^t_{-i}}(G|x_i)$, for example simply by uniform averaging of the G values in the samples associated with each x_i. Note that the single set of samples can be used no matter how many agents are in the system; we don't need a different Monte Carlo process for each agent to estimate their separate $\mathbb{E}_{q^t_{-i}}(G|x_i)$.

All agents (variables) sample moves (variable settings) independently, and coupling occurs only in the updates of the q_i. As we have seen this update (even to second order) for agent i depends only on the conditional expectations $\mathbb{E}_{q_{-i}}(G|x_i)$ where q_{-i} describes the strategies used by the other agents. Thus, if we are using Monte Carlo, then the only information which needs to be communicated to each agent is the G values upon which the estimate will be based. Using these values each agent independently updates its strategy (its q_i) in a way which collectively is guaranteed to lower the Lagrangian.

If the expectation is evaluated analytically, the ith agent needs the q_j distributions for each of the j agents involved in factors in G that also involve i. For objective functions which consists of a sum of local interactions each of which individually involves only a small subset of the variables (e.g. the problems considered here), the number of agents that i needs to communicate with may be much smaller than n.

4.2 Difference utilities for faster Monte Carlo convergence

The basic Monte Carlo approach outlined above can be slow to converge in high-dimensional problems. For the problems considered in this paper this is irrelevant, since $\mathbb{E}_{q^t_{-i}}(G|x_i)$ may be efficiently calculated in closed form for all agents i and their moves x_i, so we don't need to use Monte Carlo sampling. For completeness though here we review a variant of the basic Monte Carlo approach that converges far more quickly. See (Wolpert 2003*a*, Wolpert & Bieniawski 2004*b*) for details.

Say we are at a timestep t at the end of a Monte Carlo block, and consider the simplest updating

rule. This is gradient descent updating, in which we wish to update q_i at a timestep t by having each agent i take a small step in the direction (cf. Eq. (5))

$$\mathbf{f}^{i,G} \triangleq -\left[\frac{\partial \mathcal{L}(q^t)}{\partial q_i(x_i^1)}, \cdots, \frac{\partial \mathcal{L}(q^t)}{\partial q_i(x_i^{|\mathcal{X}_i|})}\right] - \eta_i(q)\mathbf{1}$$

where $\eta_i(q)$ was defined in Eq. (7), $\mathbf{1}$ is the vector of length $|\mathcal{X}_i|$ all of whose components are 1, and $x_i^1, \cdots, x_i^{|\mathcal{X}_i|}$ are the $|\mathcal{X}_i|$ moves available to agent i. In general, there will be some error in i's estimate of that step, since it has limited information about q_{-i}^t. Presuming quadratic loss reflects quality of the update, for agent i the Bayes-optimal estimate of its update is the posterior expectation

$$\int dq^t \; P(q_{-i}^t \mid n_i)\, \mathbf{f}^{i,G}$$

where n_i is all the prior knowledge and data that i has, and the dependence of $\mathbf{f}^{i,G}$ on q_{-i}^t is implicit.[9] $P(q_{-i}^t|n_i)$ is a probability distribution over likely values of q_{-i}^t given the information n_i available to agent i.

Now agent i can evaluate $\ln q_i(x_i)$ for each of its moves x_i exactly. However to perform its update it still needs the integrals

$$\int dq^t \; P(q_{-i}^t|n_i)\, \mathbb{E}_{q_{-i}^t}(G|x_i)$$

(recall Eq. (6)). In general these integrals can be very difficult to evaluate. As an alternative, we can replace those integrals with simple maximum likelihood estimators of them, i.e., we can use Monte Carlo sampling. In this case, the prior information, n_i, available to the agent is a list, $\mathfrak{L}$, of L joint configurations $\mathbf{x}$ along with their accompanying objective values $G(\mathbf{x})$.

To define this precisely, for any function $h(\mathbf{x})$, let $\hat{\mathbf{h}}(n_i)$ be a vector of length $|\mathcal{X}_i|$ which is indexed by x_i. The x_i component of $\hat{\mathbf{h}}(n_i)$ is indicated as $\hat{h}_{x_i}(n_i)$. Each of its components is given by the information in n_i. The x_i'th such component is the empirical average of the values that h had in the L_{x_i} samples from the just-completed Monte Carlo block when agent i made move x_i, i.e.

$$\hat{h}_{x_i}(n_i) = \frac{1}{L_{x_i}} \sum_{\mathbf{x} \in \mathfrak{L}_{x_i}} h(\mathbf{x})$$

where $\mathfrak{L}_{x_i}$ is the set of $\mathbf{x}$ in $\mathfrak{L}$ whose ith component is equal to x_i, and where $L_{x_i} = |\mathfrak{L}_{x_i}|$. Given this notation, we can express the components of the gradient update step for agent i under the simple maximum likelihood estimator as the values

$$\hat{f}^{i,G}_{x_i}(n_i) = -\{\hat{G}_{x_i}(n_i) + T \ln q_i(x_i)\} \; - \; \hat{\eta}(n_i) \tag{11}$$

where

$$\hat{\eta}_i(n_i) \triangleq \frac{-1}{|X_i|} \sum_{x'_i} \{\hat{G}_{x'_i}(n_i) + T \ln q_i(x'_i)\}. \tag{12}$$

Unfortunately, often in very large systems the convergence of $\hat{G}(n_i)$ with growing L is very slow, since the distribution sampled by the Monte Carlo process to produce n_i is very broad. This suggests we use some alternative estimator. Here we focus on estimators that are still maximum likelihood, just with a different choice of utility. To that end, first posit that the differences $\mathbb{E}_{q^t_{-i}}(G|x_i) - \mathbb{E}_{q^t_{-i}}(G|x'_i)$, one for each (x_i, x'_i) pair, are unchanged when one replaces G with some other function g_i. So the change is equivalent to adding a constant to G, as far as those differences are concerned. This means that if agent i used q^t_{-i} to evaluate its expectation values exactly, then its associated update would be unchanged under this replacement. (This is due to cancellation of the equivalent additive constant with the change that arises in $\eta_i(n_i)$ under the replacement of $G(\mathbf{x})$ with $g^i(\mathbf{x})$). It is straight-forward to verify that the set of all g^i guaranteed to have this character, regardless of the form of q, is the set of *difference utilities*, $g^i(x) = G(x) - D^i(\mathbf{x}_{-i})$ for some function D^i. G itself is the trivial case $D^i(\mathbf{x}_{-i}) = 0 \; \forall x_i$.

On the other hand, if we use a difference utility rather than G in our maximum likelihood estimator then it is the sample values of $P(g^i)$ that generate n_i, and we use the associated x_i-indexed vector $\hat{g}^i_{x_i}(n_i)$ rather than $\hat{G}_{x_i}(n_i)$ to update each q_i. For well-chosen D^i it may typically be the case that $\hat{g}^i(n_i)$ has a far smaller standard deviation than does $\hat{G}(n_i)$. In particular, if the number of coordinates coupled to i through G does not grow as the system does, often such difference utility error bars will not grow much with system size, whereas the error bars associated with G will grow greatly. Another advantage of difference utilities is that very often the Monte Carlo values of a difference utility are far easier to evaluate than are those of G, due to cancellation in subtracting D^i.

To make this more precise we can solve for the difference utility with minimal error bar. First as a notational matter, extend the definition of $\mathbf{f}^{i,G}$ by replacing G with (arbitrary) h throughout, writing that extended version as $\mathbf{f}^{i,h}$. Then assuming no $L_{x_i} = 0$, we are interested in the g^i minimizing the data-averaged quadratic error,

$$\mathbb{E}_{q_{-i},n_i^{g^i}}(\text{Err}) = \int dq_{-i}\, P(q_{-i}) \int dn_i^{g^i}\; P(n_i^{g^i}|n_i^x, q_{-i}, g^i) \|\mathbf{f}^{i,g^i} - \hat{\mathbf{f}}^{i,g^i}(n_i)\|^2, \tag{13}$$

where $P(q_{-i})$ reflects any prior information we might have concerning q_{-i} (e.g., that it is likely that the current $\mathbf{f}^{i,g^i}$ is close to that estimated for the previous block of L steps), and $n_i^{g^i}$ is the set of values of the private utility contained in n_i. (The associated x_i values, n_i^x, are independent of g^i and q_{-i} and therefore for our purposes can be treated as though they are fixed.)

Now the components of $\hat{\mathbf{f}}^{i,g^i}(n_i)$ (one for each x_i) are not independent in general, being coupled via $\hat{\eta}_i(n_i)$. To get an integrand that involves only independent variables, we must work with only one x_i component at a time. To that end, rewrite the data-averaged quadratic error as

$$\sum_{x_i} \int dq_{-i}\, P(q_{-i}) \int dn_i^{g^i}\; P(n_i^{g^i}|n_i^x, q_{-i}, g^i)[f_{x_i}^{i,g^i} - \hat{f}_{x_i}^{i,g^i}(n_i)]^2$$

where $f_{x_i}^{i,g^i}$ is the $q_i(x_i)$ component of $\mathbf{f}^{i,g^i}$. Our results will hold for all q_{-i}, so we ignore the outer integral and focus on

$$\sum_{x_i} \int dn_i^{g^i}\; P(n_i^{g^i}|n_i^x, q_{-i}, g^i)[f_{x_i}^{i,g^i} - \hat{f}_{x_i}^{i,g^i}(n_i)]^2. \tag{14}$$

For any x_i the inner integral can be decomposed with the famous bias-variance decomposition into a sum of two terms (Duda, Hart & Stork 2000).[10] The first of the two terms in our sum is the (square of the) *bias*, $f_{x_i}^{i,g^i} - \mathbb{E}_{n_i^{g^i}}(\hat{f}_{x_i}^{i,g^i})$, where

$$\mathbb{E}_{n_i^{g^i}}(\hat{f}_{x_i}^{i,g^i}(n_i^{g^i})) \triangleq \int dn_i^{g^i}\; P(n_i^{g^i}|n_i^x, q_{-i}, g^i) \hat{f}_{x_i}^{i,g^i}(n_i) \tag{15}$$

is the expectation (over all possible sets of Monte Carlo sample utility values n_i^g) of $\hat{f}_{x_i}^{i,g^i}(n_i)$. The bias reflects the systematic trend of our sample-based estimate of $f_{x_i}^{i,g^i}$ to differ from the actual $f_{x_i}^{i,g^i}$. When bias is zero, we know that on average our estimator will return the actual value it's estimating.

The second term in our sum is the *variance*,

$$\mathrm{Var}(\hat{f}^{i,g^i}_{x_i}(n_i^{g^i})) \triangleq \int dn_i^{g^i}\ P(n_i^{g^i}|n_i^x, q_{-i}, g^i)\ \{\hat{f}^{i,g^i}_{x_i}(n_i) - \mathbb{E}_{n_i^{g^i}}(\hat{f}^{i,g^i}_{x_i})\}^2, \tag{16}$$

In general variance reflects how much the value of our estimate "bounces around" if one were to resample our Monte Carlo block. In our context, it reflects how much the private utility of agent i depends on its own move x_i versus the moves of the other agents. When i's estimator is isolated from the moves of the other agents $\hat{f}^{i,g^i}_{x_i}(n_i)$ is mostly independent of the moves of the other agents, and therefore of n_i. This means that variance is low, and there is a crisp "signal-to-noise" guiding i's updates. In this situation the agent can achieve a preset accuracy level in its updating with a minimal total number of samples in the Monte Carlo block.

Plug the general form for a difference utility into the formula for $\hat{f}^{i,g^i}_{x_i}(n_i)$ to see that (due to cancellation with the $\hat{\eta}(n_i)$ term) its $n_i^{g^i}$-averaged value is independent of D^i. Accordingly bias must equal 0 for difference utilities. (In fact, difference utilities are the only utility that is guaranteed to have zero bias for all q_{-i}.) So our expected error reduces to the sum over all x_i of the variance for each x_i.

For each one of those variances again use Eq. 11 with G replaced by g^i throughout to expand $\hat{f}^{i,g^i}_{x_i}(n_i)$. Since the $q_i(x_i)$ terms in that expansion are all the same constant independent of n_i, they don't contribute to the variance. Accordingly we have

$$\begin{aligned}
\mathrm{Var}(\hat{f}^{i,g^i}_{x_i}(n_i^{g^i})) &= \mathrm{Var}\left(\hat{g}^i_{x_i}(n_i^g) - \frac{1}{|\mathcal{X}_i|}\sum_{x_i'}\hat{g}^i_{x_i'}(n_i^g)\right) \\
&= \mathrm{Var}\left(\left[1 - \frac{1}{|\mathcal{X}_i|}\right]\hat{g}^i_{x_i}(n_i^g) - \frac{1}{|\mathcal{X}_i|}\sum_{x_i' \neq x_i}\hat{g}^i_{x_i'}(n_i^g)\right).
\end{aligned} \tag{17}$$

Since n_i^x is fixed and we are doing IID sampling, the two expressions inside the variance function are statistically independent. In addition, the variance of a difference of independent variables is

the sum of the variances. Accordingly, the sum over all x_i of our variances is (cf. Eq. (14))

$$\begin{aligned}
\sum_{x_i} \mathrm{Var}\big(\hat{f}^{i,g^i}_{x_i}(n^g_i)\big) &= \sum_{x_i} \mathrm{Var}\left(\left[1 - \frac{1}{|\mathcal{X}_i|}\right]\hat{g}^i_{x_i}(n^g_i) - \frac{1}{|\mathcal{X}_i|}\sum_{x'_i \neq x_i} \hat{g}^i_{x'_i}(n^g_i)\right) \\
&= \sum_{x_i}\left\{\left[1 - \frac{1}{|\mathcal{X}_i|}\right]^2 \mathrm{Var}\big(\hat{g}^i_{x_i}(n^g_i)\big) + \frac{1}{|\mathcal{X}_i|^2}\sum_{x'_i \neq x_i} \mathrm{Var}\big(\hat{g}^i_{x'_i}(n^g_i)\big)\right\} \\
&= \sum_{x_i}\left\{\left[1 - \frac{1}{|\mathcal{X}_i|}\right]^2 + \frac{|\mathcal{X}_i| - 1}{|\mathcal{X}_i|^2}\right\} \mathrm{Var}(\hat{g}^i_{x_i}(n^g_i)) \\
&= \frac{|\mathcal{X}_i| - 1}{|\mathcal{X}_i|}\sum_{x_i} \mathrm{Var}(\hat{g}^i_{x_i}(n^g_i)), \qquad (18)
\end{aligned}$$

where the third equation follows from the second by using the trivial identity $\sum_a \sum_{b \neq a} F(b) = \sum_a F(a) \sum_{b \neq a} 1$ for any function F.

Since for each such x_i we are doing L_{x_i}-fold IID sampling of an associated fixed distribution, the variance for each separate x_i is of the form

$$\int d\mathbf{y}\, P(\mathbf{y})\left[\frac{1}{L_{x_i}}\sum_{j=1}^{L_{x_i}} y_j - \mathbb{E}_P\left(\frac{1}{L_{x_i}}\sum_{j=1}^{L_{x_i}} y_j\right)\right]^2$$

for a fixed distribution $P(y_1, y_2, \ldots, y_{L_{x_i}}) = \prod_{j=1}^{L_{x_i}} P(y_j)$. We can again use the decomposition of a variance of a sum into a sum of variances to evaluate this. With the distribution q^t implicit, define the single sample variance for the value of any function $H(x)$, for move x_i, as

$$\mathrm{Var}(H(x_i)) \triangleq \mathbb{E}([H]^2|x_i) - [\mathbb{E}(H|x_i)]^2. \qquad (19)$$

This gives

$$\mathrm{Var}(\hat{g}^i_{x_i}(n^g_i)) = \mathrm{Var}(g^i(x_i)) \,/\, L_{x_i} \qquad (20)$$

Collecting terms, we get

$$\sum_{x_i} \mathrm{Var}(\hat{f}^{i,g^i}_{x_i}(n^g_i)) = \frac{|\mathcal{X}_i| - 1}{|\mathcal{X}_i|}\sum_{x_i} \frac{\mathrm{Var}(g^i(x_i))}{L_{x_i}}. \qquad (21)$$

Now $\mathrm{Var}(A(\tau)) = (1/2)\sum_{t_1,t_2} P(t_1)P(t_2)[A(t_1) - A(t_2)]^2$ for any random variable τ with dis-

tribution $P(t)$. Use this to rewrite the sum in Eq. (21) as

$$\frac{|\mathcal{X}_i| - 1}{2|\mathcal{X}_i|} \sum_{x_i} L_{x_i}^{-1} \sum_{\mathbf{x}'_{-i}, \mathbf{x}''_{-i}} q_{-i}(\mathbf{x}'_{-i}) q_{-i}(\mathbf{x}''_{-i}) [g^i(x_i, \mathbf{x}'_{-i}) - g^i(x_i, \mathbf{x}''_{-i})]^2$$

Bring the sum over x_i inside the other integrals, expand g^i, and drop the overall multiplicative constant to get

$$\begin{aligned} \mathrm{Var}(\hat{F}^i_{g^i}(n^g_i)) \;\propto\; & \sum_{\mathbf{x}'_{-i}, \mathbf{x}''_{-i}} q_{-i}(\mathbf{x}'_{-i}) q_{-i}(\mathbf{x}''_{-i}) \\ & \sum_{x_i} \frac{\left[G(x_i, \mathbf{x}'_{-i}) - G(x_i, \mathbf{x}''_{-i}) - \{D^i(\mathbf{x}'_{-i}) - D^i(\mathbf{x}''_{-i})\}\right]^2}{L_{x_i}}. \end{aligned}$$

For each $\mathbf{x}'_{-i}$ and $\mathbf{x}''_{-i}$, our choice of D^i minimizes the sum so long as the difference in the curly brackets obeys

$$D^i(\mathbf{x}'_{-i}) - D^i(\mathbf{x}''_{-i}) = \sum_{x_i} \frac{[G(x_i, \mathbf{x}'_{-i}) - G(x_i, \mathbf{x}''_{-i})]}{L_{x_i}} \Big/ \sum_{x'_i} \frac{1}{L_{x'_i}}.$$

This can be assured by picking

$$D^i(\mathbf{x}_{-i}) = \left[\sum_{x'_i} \frac{1}{L_{x'_i}}\right]^{-1} \sum_{x'_i} \frac{G(x'_i, \mathbf{x}_{-i})}{L_{x'_i}}. \tag{22}$$

for all $\mathbf{x}_{-i}$. The associated difference utility, $g^i(\mathbf{x}) = G(\mathbf{x}) - D^i(\mathbf{x}_{-i})$, is called the *Aristocrat utility* (AU). An approximation to it was investigated in (Wolpert & Tumer 2001, Wolpert, Tumer & Bandari 2002, Wolpert & Tumer 2002) and references therein. AU itself was derived in (Wolpert 2003*b*).

Note that AU minimizes variance of gradient descent updating *regardless of the form of* q. Indeed, being independent of q_{-i}, it minimizes our original q_{-i} integral in Eq. (13), regardless of the prior $P(q_{-i})$. For the same reason it is optimal if the integral is replaced by a worst-case bound over q_{-i}.

Sometimes not all the terms in the sum in AU can be stored, because $|\mathcal{X}_i|$ and/or the block size is too large. In such a case n^x_i must be averaged over as well as n^g_i. That sum can be approximated

by replacing the L_{x_i} values in the definition of AU with $q(x_i)L$. This replacement also provides a way to address cases where one or more $L_{x_i} = 0$.[11] Similarly, for computational reasons it may be desirable to approximate the weighted average of G over all x'_i which defines AU.

The sum over x'_i occurring in AU should not be confused with the sum over x'_i that pulls our gradient estimate back into the unit simplex. The sum here is over values of G for counterfactuals sample pairs $(x'_i, \mathbf{x}_{-i})$. (The other sum is over values of our gradient estimate at all of its arguments.) When the functional form of G is known it is often the case that there is cancellation which allows AU be calculated directly, in one evaluation, an evaluation which can be cheaper than that of $G(\mathbf{x})$. When this is not the case their evaluation incurs a computational cost in general.[12] This cost is offset by the fact that those evaluations allow us to determine the value of AU not just for the actual point $(x_i, \mathbf{x}_{-i})$, but in fact for all points $\{(x'_i, \mathbf{x}_{-i}) | x'_i \in \mathcal{X}_i\}$.

Nonetheless, there will be cases where evaluating AU requires evaluating all possible $G(x'_i, \mathbf{x}_{-i})$, and where the cost of that is prohibitive, even if it allows us to update AU for all x_i at once. Fortunately there are difference utilities that are cheaper to evaluate than AU while still having less variance than G. In particular, note that the weighting factor $L_{x'_i}^{-1} / \sum_{x''_i} L_{x''_i}^{-1}$ in the formula for AU is largest for those x_i which occur infrequently, i.e. that have low $q_i(x_i)$. This observation leads to the *Wonderful Life Utility* (WLU), which is an approximation to AU that (being a difference utility) also has zero bias:

$$g_i^{WLU}(x_i, \mathbf{x}_{-i}) = G(x_i, \mathbf{x}_{-i})) - G(x_i^{\text{clamp}}, \mathbf{x}_{-i}). \tag{23}$$

In this formula, $x_i^{\text{clamp}} = \arg\min_{x_i} L_{x_i}$ or if we wish to be more conservative, $\arg\min_{x_i} q_i(x_i)$, agent i's lowest probability move (Wolpert 2003*a*, Wolpert 2004*a*).[13]

4.3 Discussion of Monte Carlo sampling

Note that the foregoing analysis breaks down if any of the $L_{x_i} = 0$. More generally it may break down if just one or more of the $q(x_i)$ are particularly small in comparison to the others, even if no L_{x_i} is exactly zero. The reason for this is that our approximation of the average over n_i^x with the average where no $L_{x_i} = 0$ breaks down. Doing the exact calculation with no such approximation doesn't fix the situation — once we have to assign non-infinitesimal probability to $L_{x_i} = 0$, we're allowing

a situation in which the gradient step would take us off the simplex of allowed $q \in \mathcal{Q}$. We might try to compensate for this by reducing the stepsize, but in general the foregoing analysis doesn't hold if stepsize is reduced in some situations but not in any others. (Variable stepsize constitutes a change to the update rule. Such a modification to the update rule must be incorporated into the analysis — which obviates the derivation of AU.)

One way to address this scenario would be to simply zero out the probability of agent i making any move x_i for which $q_i(x_i)$ is particular small. In other words, we can redefine i's move space to exclude any moves if their probability ever gets sufficiently small. This has the additional advantage of reducing the amount of "noise" that agents $j \neq i$ will see in the next Monte Carlo block, since the effect of agent i on the value of G in that block is more tightly constrained.

There several ways to extend the derivation of AU, which only addresses estimation error for a single agent at a time, and for just that agent's current update. One such extension is to have agent i's utility set to improve the accuracy of the update estimation for agents $j \neq i$. For example, we could try to bias q_i to be peaked about only a few moves, thereby reducing the amount of noise those other agents $j \neq i$ will see in the next Monte Carlo block due to variability in i's move choice. Another extension is to have agent i's utility set to improve the accuracy of its estimate of its update for future Monte Carlo blocks, even at the expense of accuracy for the current block..

Strictly speaking, the derivation of AU only applies to gradient descent updating of q. Difference utilities are unbiased estimators for the Nearest Newton update rule, so long as each i estimates $\mathbb{E}_{q^t}(g^i)$ as $q_i^t(x_i)$ times the estimate of $\mathbb{E}_{q^t}(g^i|x_i)$,

$$\sum_{x_i} \hat{g}^i_{x_i}(n_i) q_i^t(x_i)$$

rather than as the empirical average over all samples of g^i,[14]

$$\sum_{x_i} \hat{g}^i_{x_i}(n_i) \frac{L_{x_i}}{L}.$$

However the calculation for how to minimize the variance must be modified. Redoing the algebra

above, the analog of AU for the Nearest Newton rule arises if we replace

$$\frac{1}{L_{x_i}} \rightarrow \frac{[q_i(x_i)]^2}{L_{x_i}} \left\{ [1 - q_i(x_i)]^2 + \sum_{x'_i \neq x_i} [q_i(x'_i)]^2 \right\} \tag{24}$$

throughout the equation defining AU. Similar considerations apply to Brouwer updating as well. Nonetheless, in practice AU and WLU as defined above work well (and in particular far better than taking $g^i = G$) for the other updating rules as well.

For gradient descent updating, minimizing expected quadratic error of our estimator of $\mathbb{E}_{q^t_{-i}}(G|x_i)$ corresponds to making a quadratic approximation to the Lagrangian surface, and then minimizing the expected value of the Lagrangian after the gradient step (Wolpert 2003*a*). More generally, and especially for other update rules, some other kind of error measure might be preferable. Such measures would differ from the bias-variance decomposition. We do not consider such alternatives here.

Note that the agents are completely "blind" in the Monte Carlo process outline above, getting no information from other agents other than the values of $G(\mathbf{x})$. When we allow some information to be transmitted between the agents we can improve the estimation of $\mathbb{E}_{q^t_{-i}}(G|x_i)$ beyond that of the simple Monte Carlo process outlined above. For example, say that at every timestep the agent i knows not just its own move x_i, but in fact the joint move $\mathbf{x}$. Then as time goes on it accumulates a training set of pairs $\{(\mathbf{x}, G(\mathbf{x}))\}$. These can be used with conventional supervised learning algorithms (Duda et al. 2000) to form a rough estimate of the entire function G, $\hat{G}$. Say that in addition i knows not its own distribution $q_i(x^t_i)$, but in fact the entire joint distribution, $q(\mathbf{x}^t)$. Then it can use that joint distribution together with $\hat{G}$ to form an estimate of $\mathbb{E}_{q^t_{-i}}(G|x_i)$. That estimate is in addition to the one formed by the blind Monte Carlo process outlined above. One can then combine these estimates to form one superior to both. See (Lee & Wolpert 2004).

Even when we are restricted to a blind Monte Carlo process, there are many heuristics that when incorporated into the update rules that can greatly improve their performance on real-world problems (Wolpert 2004*c*). In this paper we examine problems for which joint distributions q are known to all agents as well as the function form of $G(\mathbf{x})$ and the required expectations $\mathbb{E}_{q^t_{-i}}(G|x_i)$ may be obtained in closed form. So there is no need for Monte Carlo approximations. Accordingly there is no need for those heuristics, and there is not even any need to using difference utilities.

Empirical investigations of the effects of using difference utility functions and the heuristics may be found in (Bieniawski & Wolpert 2004*a*, Bieniawski et al. 2004, Bieniawski & Wolpert 2004*b*).

5 Semicoordinate Transformations

5.1 Motivation

Consider a multi-stage game like chess, with the stages (i.e., the instants at which one of the players makes a move) delineated by t. In game theoretic terms, the "strategy" of a player is the mapping from board-configuration to response that specifies the rule it adopts before play starts (Fudenberg & Tirole 1991, Basar & Olsder 1999, Osborne & Rubenstein 1994, Aumann & Hart 1992, Fudenberg & Levine 1998). More generally, in a multi-stage game like chess the strategy of player i, x_i, is the set of t-indexed maps taking what that player has observed in the stages $t' < t$ into its move at stage t. Formally, this set of maps is called player i's **normal form** strategy.

The joint strategy of the two players in chess sets their joint move-sequence, though in general the reverse need not be true. In addition, one can always find a joint strategy to result in any particular joint move-sequence. Now typically at any stage there is overlap in what the players have observed over the preceding stages. This means that even if the players' strategies are statistically independent (being separately set before play started), their move sequences are statistically coupled. In such a situation, by parameterizing the space $\mathcal{Z}$ of joint-move-sequences $\mathbf{z}$ with joint-strategies $\mathbf{x}$, we shift our focus from the coupled distribution $P(\mathbf{z})$ to the decoupled product distribution, $q(\mathbf{x})$. This is the advantage of casting multi-stage games in terms of normal form strategies.

More generally, given any two spaces $\mathcal{X}$ and $\mathcal{Z}$, any associated onto mapping $\zeta : \mathcal{Z} \rightarrow \mathcal{X}$, not necessarily invertible, is called a *semicoordinate system*. The identity mapping $\mathcal{Z} \rightarrow \mathcal{Z}$ is a trivial example of a semicoordinate system. Another semicoordinate system is the mapping from joint-strategies in a multi-stage game to joint move-sequences. In other words, changing the representation space of a multi-stage game from move-sequences $\mathbf{z}$ to strategies $\mathbf{x}$ is a semicoordinate transformation of that game.

Intuitively, a semi-coordinate transformation is a reparameterization of how a game — a mapping from joint moves to associated payoffs — is represented. So we can perform a semicoordinate

transformation even in a single-stage game. Say we restrict attention to distributions over $\mathcal{X}$ that are product distributions. Then changing $\zeta(\cdot)$ from the identity map to some other function means that the players' moves are no longer independent. After the transformation their move choices — the components of z — are statistically coupled, even though we are considering a product distribution.

Formally, this is expressed via the standard rule for transforming probabilities,

$$P_{\mathcal{Z}}(\mathbf{z} \in \mathcal{Z}) \triangleq \zeta(P_{\mathcal{X}}) \triangleq \int d\mathbf{x}\, P_{\mathcal{X}}(\mathbf{x})\delta(\mathbf{z} - \zeta(\mathbf{x})), \tag{25}$$

where $P_{\mathcal{X}}$ and $P_{\mathcal{Z}}$ are the distributions across $\mathcal{X}$ and $\mathcal{Z}$, respectively. To see what this rule means geometrically, recall that $\mathcal{P}$ is the space of all distributions (product or otherwise) over $\mathcal{Z}$ and that $\mathcal{Q}$ is the space of all product distributions over $\mathcal{X}$. Let $\zeta(\mathcal{Q})$ be the image of $\mathcal{Q}$ in $\mathcal{P}$. Then by changing $\zeta(\cdot)$, we change that image; different choices of $\zeta(\cdot)$ will result in different manifolds $\zeta(\mathcal{Q})$.

As an example, say we have two players, with two possible moves each. So $\mathbf{z}$ consists of the possible joint moves, labelled $(0,0), (0,1), (1,0)$ and $(1,1)$. Have $\mathcal{X} = \mathcal{Z}$, and choose $\zeta(0,0) = (0,0)$, $\zeta(0,1) = (1,1)$, $\zeta(1,0) = (1,0)$, and $\zeta(1,1) = (0,1)$. Say that q is given by $q_1(x_1 = 0) = q_2(x_2 = 0) = 2/3$. Then the distribution over joint-moves $\mathbf{z}$ is $P_{\mathcal{Z}}(0,0) = P_{\mathcal{X}}(0,0) = 4/9$, $P_{\mathcal{Z}}(1,0) = P_{\mathcal{Z}}(1,1) = 2/9$, $P_{\mathcal{Z}}(0,1) = 1/9$. So $P_{\mathcal{Z}}(\mathbf{z}) \neq P_{\mathcal{Z}}(z_1)P_{\mathcal{Z}}(z_2)$; the moves of the players are statistically coupled, even though their strategies x_i are independent.

Any $P_{\mathcal{Z}}$, no matter what the coupling among its components, can be expressed as $\zeta(P_{\mathcal{X}})$ for some product distribution $P_{\mathcal{X}}$ for and associated $\zeta(\cdot)$ In the worst case, one can simply choose $\mathcal{X}$ to have a single component, with $\zeta(\cdot)$ a bijection between that component and the vector z — trivially, any distribution over such an $\mathcal{X}$ is a product distribution. Another simple example is where one aggregates one or more agents into a new single agent, i.e., replaces the product distribution over the joint moves of those agents with an arbitrary distribution over their joint moves. This is related to the concept coalitions in cooperative game theory, as well as to Aumann's correlated equilibrium (Fudenberg & Tirole 1991, Aumann 1987, Aumann & Hart 1992).

Less trivially, given any model class of distributions $\{P_{\mathcal{Z}}\}$, there is an $\mathcal{X}$ and associated $\zeta(\cdot)$ such that $\{P_{\mathcal{Z}}\}$ is identical to $\zeta(\mathcal{Q}_X)$. Formally this is expressed in a result concerning Bayes nets. For simplicity, restrict attention to finite $\mathcal{Z}$. Order the components of $\mathcal{Z}$ from 1 to N. For each index

$i \in \{1, 2, \ldots, N\}$, have the parent function $\mathrm{Pa}(i, \mathbf{z})$ fix a subset of the components of $\mathbf{z}$ with index greater than i, returning the value of those components for the $\mathbf{z}$ in its second argument if that subset of components is non-empty. So for example, with $N > 5$, we could have $\mathrm{Pa}(1, \mathbf{z}) = (z_2, z_5)$. Another possibility is that $\mathrm{Pa}(1, \mathbf{z})$ is the empty set, independent of $\mathbf{z}$.

Let $A(\mathrm{Pa})$ be the set of all probability distributions $P_{\mathcal{Z}}$ that obey the conditional dependencies implied by Pa: $\forall\, P_{\mathcal{Z}} \in A(\mathrm{Pa}), \mathbf{z} \in \mathcal{Z}$,

$$P_{\mathcal{Z}}(\mathbf{z}) \;=\; \prod_{i=1}^{N} P_{\mathcal{Z}}(z_i | \mathcal{P}(i, \mathbf{z})). \tag{26}$$

By definition, if $\mathrm{Pa}(i, \mathbf{z}))$ is empty, $P_{\mathcal{Z}}(z_i | \,\mathrm{Pa}(i, \mathbf{z}))$ is just the i'th marginal of $P_{\mathcal{Z}}$, $P_{\mathcal{Z}}(z_i)$. As an example of these definitions, the dependencies $\{\mathrm{Pa}(1, \mathbf{z}) = (z_2, z_3), \mathrm{Pa}(2, \mathbf{z}) = z_4, \mathrm{Pa}(3, \mathbf{z}) = (), \mathrm{Pa}(4, \mathbf{z}) = ()\}$ correspond to the family of distributions factoring as

$$P(\mathbf{z}) = P(z_1 | z_2, z_3) P(z_2 | z_4) P(z_3) P(z_4)$$

As proven in (Wolpert & Bieniawski 2004*a*), for any choice of Pa there is an associated set of distributions $\zeta(\mathcal{Q}_X)$ that equals $A(\mathrm{Pa})$ exactly:

Proposition: Define the components of X using multiple indices: For all $i \in \{1, 2, \ldots, N\}$ and possible associated values (as one varies over $\mathbf{z} \in \mathcal{Z}$) of the vector $\mathrm{Pa}(i, \mathbf{z})$, there is a separate component of $\mathbf{x}$, $x_{i;\mathrm{Pa}(i,\mathbf{z})}$. This component can take on any of the values that z_i can. Define $\zeta(\cdot)$ recursively, starting at $i = N$ and working to lower i, by the following rule: $\forall\, i \in \{1, 2, \ldots, N\}$,

$$[\zeta(\mathbf{x})]_i \;=\; x_{i;\mathrm{Pa}(i,\mathbf{z})}.$$

Then $A(\mathrm{Pa}) = \zeta(\mathcal{Q}_X)$.

Intuitively, each component of $\mathbf{x}$ in Prop. 1 is the conditional distribution $P_{\mathcal{Z}}(z_i | \,\mathrm{Pa}(i, \mathbf{z}))$ for some particular instance of the vector $\mathrm{Pa}(i, \mathbf{z})$. As illustration consider again the example $\{\mathrm{Pa}(1, \mathbf{z}) = (z_2, z_3), \mathrm{Pa}(2, \mathbf{z}) = z_4, \mathrm{Pa}(3, \mathbf{z}) = (), \mathrm{Pa}(4, \mathbf{z}) = ()\}$. If each z_i assumes the value 0 or 1, then $\mathbf{x}$ has 8 components $x_4, x_3, x_{2;0}, x_{2;1}, x_{1;00}, x_{1;01}, x_{1;10}$, and $x_{1;11}$ with each component also

$x_1 = 0$	(0,0,0), (0,0,1), (0,1,0), (0,1,1)	$z_1 = 0$	(1,0,0), (0,1,0), (0,0,1), (1,1,1)
$x_1 = 1$	(1,0,0), (1,0,1), (1,1,0), (1,1,1)	$z_1 = 1$	(0,0,0), (1,1,0), (1,0,1), (0,1,1)
$x_2 = 0$	(0,0,0), (0,0,1), (1,0,0), (1,0,1)	$z_2 = 0$	(1,0,0), (0,1,0), (1,0,1), (0,1,1)
$x_2 = 1$	(0,1,0), (0,1,1), (1,1,0), (1,1,1)	$z_2 = 1$	(0,0,0), (1,1,0), (0,0,1), (1,1,1)
$x_3 = 0$	(0,0,0), (0,1,0), (1,0,0), (1,1,0)	$z_3 = 0$	(1,0,0), (0,1,0), (1,0,1), (0,1,1)
$x_3 = 1$	(0,0,1), (0,1,1), (1,0,1), (1,1,1)	$z_3 = 1$	(0,0,0), (1,1,0), (0,0,1), (1,1,1)

Table 1: Resultant partitions from the transformation of Figure 1(b).

either 0 or 1. The product distribution in $\mathcal{X}$ is

$$q(\mathbf{x}) = q_4(x_4)q_3(x_3)q_{2;0}(x_{2;0})q_{2;1}(x_{2;1})q_{1;00}(x_{1;00})q_{1;01}(x_{1;01})q_{1;10}(x_{1;10})q_{1;11}(x_{1;11}).$$

Under ζ the distribution $q_4(x_4)$ is mapped to $q_4(z_4)$, $q_{2;0}(x_{2;0})$ is mapped to $q_2(z_2|z_4 = 0)$, $q_{1;01}(x_{1;01})$ is mapped to $q_1(z_1|z_2 = 0, z_3 = 1)$, and so on.

Prop. 1 means that in principle we never need consider coupled distributions. It suffices to restrict attention to product distributions, so long as we use an appropriate semicoordinate system. As we shall see, mixture models over $\mathcal{Z}$ can be also be represented using products. However, before discussing mixture models we show how transformation of semicoordinate systems can in principle be used to escape local minima in $L(q)$.

5.2 Semicoordinate transformations and local minima

To illustrate another application of semicoordinate transformations, we confine ourselves to the case where $\mathcal{X} = \mathcal{Z}$ so that ζ is a bijection on $\mathcal{X}$.

We assume that the domain of the ith of n variables has size $|\mathcal{X}_i|$. Then $|\mathcal{X}| = \prod_{i=1}^{n} |\mathcal{X}_i|$ is the size of the search space. Each coordinate variable x_i partitions the search space into $|\mathcal{X}_i|$ disjoint regions. The partitions are such that the intersection over all variable coordinates yields a single $\mathbf{x}$. In particular, the standard semicoordinate system relies on the partition $[*, \cdots, *, x_i = 0, *, \cdots, *]$, $\cdots$, $[*, \cdots, *, x_i = |\mathcal{X}_i| - 1, *, \cdots, *]$ for each coordinate $\mathbf{x}_i$.

As a illustrative example, consider 3 binary variables where $\mathcal{X} = \{0, 1\}^3$. Figure 1(a) shows the 8 points in the search space represented in the standard coordinate system. Figure 1(b) shows a shuffling of the 8 configurations under the permutation $(0\,1\,2\,3\,4\,5\,6\,7) \xrightarrow{\zeta} (1\,5\,2\,6\,4\,0\,7\,3)$. The resulting partitions of configurations are given in Table 1.

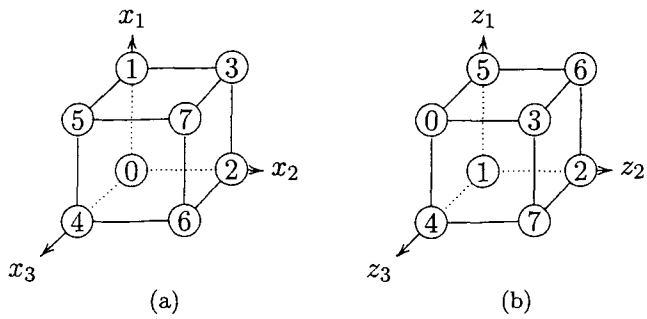


Figure 1: (a) Original linear indexing for 3 binary variables x_1, x_2, x_3. (b) Result after applying the transformation to the new variables z_1, z_2, z_3.

Such transformations can be used to escape from local minima of the Lagrangian. To see this consider a coordinate transformation ζ from $\mathcal{X}$ to the new space $\mathcal{Z}$ such that $\mathbf{z} = \zeta(\mathbf{x})$, and choose $q(\mathbf{z}) = q(\mathbf{x})$ (i.e. do not change the associated probabilities). Then the entropy contribution to the Lagrangian remains unchanged, but the expected G alters from $\sum_{\mathbf{x}} G(\mathbf{x})q(\mathbf{x})$ to[15]

$$\sum_{\mathbf{x}} G_{\mathcal{X}}(\mathbf{x})q(\mathbf{x}) \triangleq \sum_{\mathbf{x}} G(\zeta(\mathbf{x}))q(\mathbf{x}) = \sum_{\mathbf{z}} G(\zeta(\mathbf{z}))q(\mathbf{z}).$$

This means that the gradient of the maxent Lagrangian will typically differ before and after the application of ζ. In particular, what was a local minimum with zero gradient before the semicoordinate transformation may not be a local minimum after the transformation and the resultant shuffling of utility values. As difficult problems typically have many local minima in their Lagrangian, such semicoordinate transformations may prove very useful.

A simple example is shown in 2(a) where a Lagrangian surface for 2 binary variables is shown. The utility values are $G(0,0) = 0, G(1,0) = 25, G(0,1) = 18, G(1,1) = 2$. If the temperature is 7 in units of the objective then the global minimum is at $q_1(0) = 0.95, q_2(0) = 0.91$ where $L = -0.82$. At this temperature there is a suboptimal local minimum (indicated by the dot in the lower left) located at $q_1^*(0) = 0.14, q_2^*(0) = 0.08$ where $L = 0.83$.

There are a number of criteria that might be used to determine a semicoordinate transformation to escape from this local minimum q^*. Two simple choices are to select the transformation that minimizes the new value of the maxent Lagrangian (i.e., minimize $L(q^*)$), or to select the transformation which results in the largest gradient of the maxent Lagrangian at q^*, (i,e,, maxi-

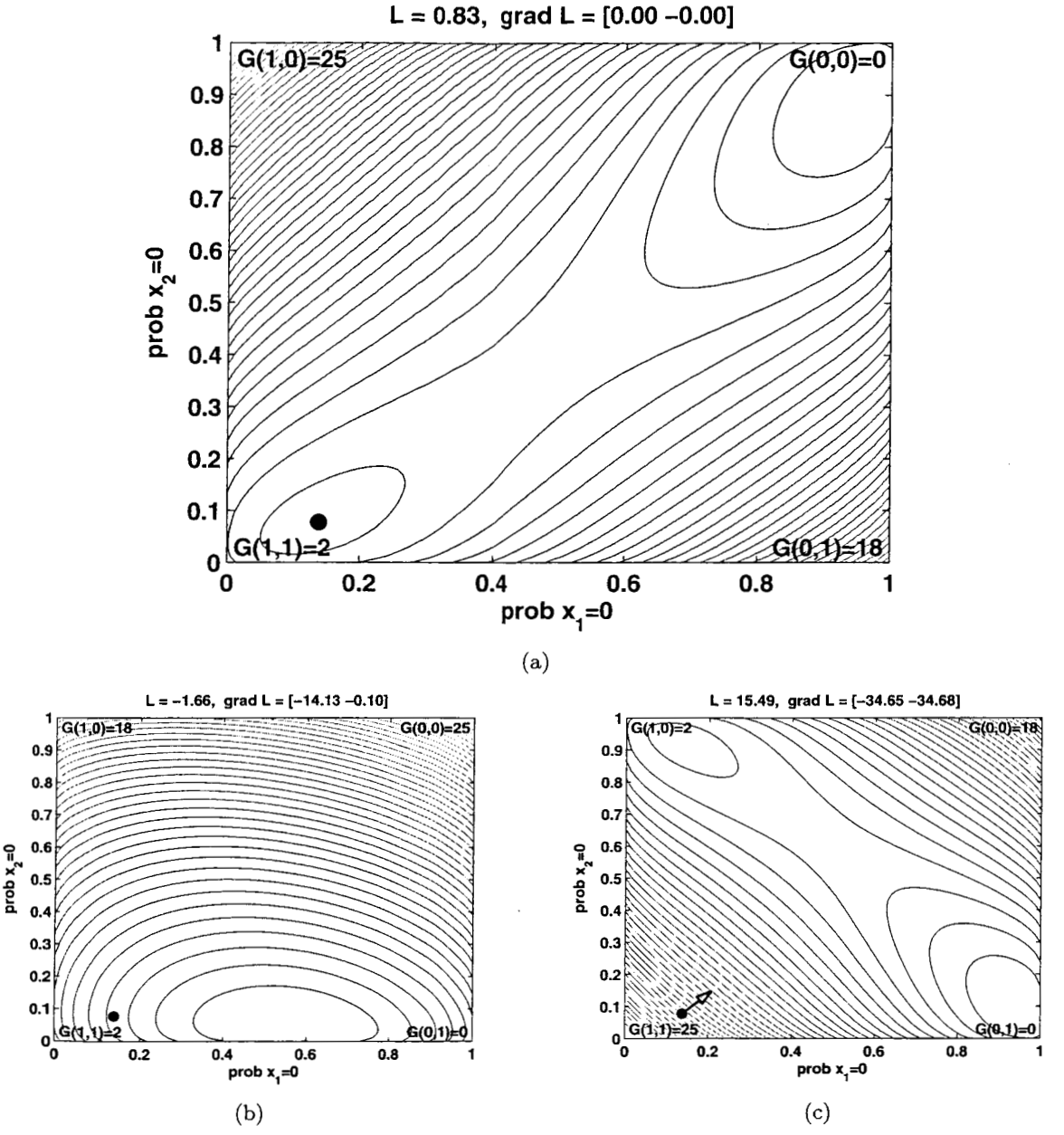


Figure 2: (a) Original Lagrangian function. The suboptimal local minimum is located at q^* which is indicated with a dot in the lower left corner. (b) The Lagrangian under the coordinate transformation which minimizes $L(q^*)$. (c) The Lagrangian under the coordinate transformation which maximizes the norm of the gradient at q^*. The direction of the negative gradient is indicated by a black arrow.

mize $\|\nabla_q L(q^*)\|$). For this simple problem the results of both these choices are shown as Figures 2(b) and 2(c) respectively. The transformation in each of these cases can be determined from the shuffling of G values.

5.3 Semicoordinate Transformations for Mixture Distributions

We have described how the Lagrangian measuring the distance of a product (mean field) distribution to a Boltzmann distribution may be minimized in a distributed fashion. We now extend these results to mixtures of product distributions, in order to represent multiple product distribution solutions at once. As mentioned above, we can always do that by means of a semicoordinate transformation of the underlying variables, allowing us to express that mixture distribution as the image of a product distribution over a different space. In this section we demonstrate this explicitly.

Let $\mathbf{x} \in \mathcal{X}$ indicate the new set of variables in a space of dimension $d_{\mathcal{X}}$, where $\mathbf{z} \in \mathcal{Z}$ is the original (pre-transformation) space over which G is defined. Then a product distribution over $\mathcal{X}$ (where $d_{\mathcal{X}} > n$, the dimension of $\mathcal{Z}$), and an appropriately chosen mapping $\zeta : \mathcal{X} \rightarrow \mathcal{Z}$ induces a mixture distribution over $\mathcal{Z}$.

To see this consider an M component mixture distribution over n variables, which we write as: $q(z) = \sum_{m=1}^{M} q^0(m) q^m(\mathbf{z})$ with $\sum_{m=1}^{M} q^0(m) = 1$ and $q^m(\mathbf{z}) = \prod_{i=1}^{n} q_i^m(z_i)$. We can express this $q(\mathbf{z})$ as (the image of) a product distribution over a space $\mathcal{X}$ of dimension $d_{\mathcal{X}} = 1 + Mn$. Intuitively, the first dimension of $\mathcal{X}$ (indicated as $x^0 \in [1, M]$) labels the mixture, and the remaining Mn dimensions (indicated as $x_i^m \in \mathcal{Z}_i$) correspond to each of the original n dimensions for each of the M mixtures.

More precisely, write out the $\mathcal{X}$-space product distribution as $q_{\mathcal{X}}(\mathbf{x}) = q^0(x^0) \prod_{m=1}^{M} q^m(\mathbf{x}^m)$ with $q^m(\mathbf{x}^m) = \prod_{i=1}^{N} q_i^m(x_i^m)$ for $\mathbf{x} = [x^0, \mathbf{x}^1, \cdots, \mathbf{x}^M]$ and $\mathbf{x}^m = [x_1^m, \cdots, x_N^m]$. The density in $\mathcal{X}$ and $\mathcal{Z}$ are related as usual by $q(\mathbf{z}) = \sum_{\mathbf{x}} q_{\mathcal{X}}(\mathbf{x}) \delta\big(\mathbf{z} - \zeta(\mathbf{x})\big)$ for our vector-valued mapping $\zeta : \mathcal{X} \rightarrow \mathcal{Z}$, with the delta function of vectors being understood component-wise. If we label the components

of ζ so that $z_i = \zeta_i(x^0, \mathbf{x}^1, \cdots, \mathbf{x}^M) \triangleq x_i^{x^0}$ we find

$$\begin{aligned} q(\mathbf{z}) &= \sum_{x^0} q^0(x^0) \sum_{\mathbf{x}^1,\cdots,\mathbf{x}^M} \prod_m q^m(\mathbf{x}^m) \prod_i \delta\big(z_i - \zeta_i(x^0, \mathbf{x}^1, \cdots, \mathbf{x}^M)\big) \\ &= \sum_{x^0} q^0(x^0) \sum_{\mathbf{x}^1,\cdots,\mathbf{x}^M} \prod_m q^m(\mathbf{x}^m) \prod_i \delta\big(z_i - x_i^{x^0}\big) \\ &= \sum_{x^0} q^0(x^0) \sum_{\mathbf{x}^{x^0}} q^{x^0}(\mathbf{x}^{x^0}) \prod_i \delta\big(z_i - x_i^{x^0}\big) \\ &= \sum_{x^0} q^0(x^0) q^{x^0}(\mathbf{z}) \end{aligned}$$

Thus, under ζ the product distribution $q_{\mathcal{X}}$ is mapped to the mixture of products $q(\mathbf{z}) = \sum_m q^0(m) q^m(\mathbf{z})$ (after relabelling x^0 to m), as desired.

The maxent Lagrangian of the $\mathcal{X}$ product distribution $q_{\mathcal{X}}(\mathbf{x})$ is

$$\mathcal{L}(q_{\mathcal{X}}) = \sum_m q^0(m) \mathbb{E}_{q^m}(G) - T\Big[S(q^0) + \sum_{m=1}^M S(q^m)\Big].$$

This Lagrangian contains a term pushing us (as we search for the minimizer of that Lagrangian) to maximize the entropy of the mixture weights. However, it provides no incentive for the distributions q^m to differ from each other.

If we wish to have the q^m differ from one another, we can instead consider the maxent Lagrangian over $q(\mathbf{z})$. In this case

$$\begin{aligned} \mathcal{L}_{\mathcal{Z}}(q) &= \sum_{\mathbf{z}} G(\mathbf{z}) q(\mathbf{z}) - TS(q) = \sum_{\mathbf{x}} G\big(\boldsymbol{\zeta}(\mathbf{x})\big) q_{\mathcal{X}}(\mathbf{x}) - TS(q) \\ &= \sum_m q^0(m) \mathbb{E}_{q^m}(G) - TS\Big(\sum_m q^0(m) q^m(\mathbf{z})\Big). \end{aligned}$$

The entropy term differs crucially in these two maxent Lagrangians. To see this add and subtract $T \sum_m q^0(m) S(q^m)$ to the $\mathcal{Z}$ Lagrangian to find

$$\mathcal{L}_{\mathcal{Z}}(q) = \sum_m q^0(m) \mathcal{L}(q^m) - TJ(q) \tag{27}$$

where each $\mathcal{L}(q^m)$ is a maxent Lagrangian as given by Eq. (1), and $J(q) \geq 0$ is a modified version

of the Jensen-Shannon (JS) distance,

$$J(q) = S\Big(\sum_m q^0(m)q^m\Big) - \sum_m q^0(m)S(q^m) = -\sum_m \sum_{\mathbf{z}} q^0(m)q^m(\mathbf{z}) \ln \frac{q(\mathbf{z})}{q^m(\mathbf{z})}.$$

Conventional Jensen Shannon distance is defined to compare two distributions to each other, and gives those distributions equal weight (Lin 1991). In contrast, the generalized JS distance $J(q)$ concerns multiple distributions, and weights them nonuniformly, according to $q^0(m)$.

$J(q)$ is maximized when the q^m are all different from each other. Thus its inclusion in our Lagrangian pushes us to have the mixing components $\{q^m(\mathbf{x})|m = 1, \ldots, M\}$ be far apart (in $\mathcal{X}$) from one another. In this, we can view Eq. (27) as a novel derivation of (a generalized version of) Jensen Shannon distance. Unfortunately, it also couples all of the variables (because of the sum inside the logarithm), preventing a highly distributed solution.

To address this, in this paper we replace $J(q)$ in $\mathcal{L}_Z(q)$ with another function which lower-bounds $J(q)$ but which requires less communication between agents. It is this modified Lagrangian that we will minimize.

5.4 A Variational Lagrangian

Following (Jaakkola & Jordan 1998), we introduce M variational functions $w(\mathbf{z}|m)$ and lower-bound the true JS distance with

$$\begin{aligned} J(q) &= -\sum_m \sum_{\mathbf{z}} q^0(m)q^m(\mathbf{z}) \ln \left[\frac{1}{w(\mathbf{z}|m)} q^0(m) \frac{w(\mathbf{z}|m)q(\mathbf{z})}{q^0(m)q^m(\mathbf{z})}\right] \\ &= \sum_m \sum_{\mathbf{z}} q^0(m)q^m(\mathbf{z}) \ln w(\mathbf{z}|m)) - \sum_m q^0(m) \ln q^0(m) \\ &\quad - \sum_m \sum_{\mathbf{z}} q^0(m)q^m(\mathbf{z}) \ln \frac{w(\mathbf{z}|m)q(\mathbf{z})}{q^0(m)q^m(\mathbf{z})}. \end{aligned}$$

Now replace M of the $-\ln$ terms with the lower bound $-\ln z \geq -\nu z + \ln \nu + 1$ obtained from the Legendre dual of the logarithm to find

$$J(q) \geq J(q, w, \nu) \triangleq \sum_m \sum_{\mathbf{z}} q^0(m) q^m(z) \ln w(\mathbf{z}|m) - \sum_m q^0(m) \ln q^0(m)$$
$$- \sum_m \nu_m \sum_{\mathbf{z}} w(\mathbf{z}|m) q(z) + \sum_m q^0(m) \ln \nu_m + 1.$$

Optimization over w and ν maximizes this lower bound. To further aid in distributing the algorithm we restrict the class of variational $w(\mathbf{z}|m)$ to products: $w(\mathbf{z}|m) = \prod_i w_i(z_i|m)$. For this choice

$$J(q, w, \nu) \triangleq \sum_m q^0(m) \left\{ B^{m,m} - \sum_{\tilde{m}} A^{m,\tilde{m}} \nu_{\tilde{m}} + \ln \nu_m \right\} + S(q^0) + 1 \tag{28}$$

where $A_i^{\tilde{m},m} \triangleq \sum_{z_i} q_i^{\tilde{m}}(z_i) w_i(z_i|m)$, $A^{\tilde{m},m} \triangleq \prod_{i=1}^d A_i^{\tilde{m},m}$, $B_i^{m,m} \triangleq \sum_{z_i} q_i^m(z_i) \ln w_i(z_i|m)$, and $B^{m,m} \triangleq \sum_{i=1}^d B_i^{m,m}$.[16] At any temperature T the variational Lagrangian which must be minimized with respect to q, w and ν (subject to appropriate positivity and normalization constraints) is then

$$\mathcal{L}_{\mathcal{Z}}(q, w, \nu) = \sum_m q^0(m) \mathcal{L}(q^m) - T J(q, w, \nu) \tag{29}$$

with $J(q, w, \nu)$ given by Eq. (28).

6 Minimizing the Mixture Distribution Lagrangian

Equating the gradients with respect to w and ν to zero gives

$$\frac{1}{\nu_m} = \frac{1}{q^0(m)} \sum_{\tilde{m}} q^0(\tilde{m}) A^{\tilde{m},m}. \tag{30}$$

$$w_i(z_i|m) \propto \frac{q^0(m) q_i^m(z_i)}{\nu_m} \left[\sum_{\tilde{m}} q^0(\tilde{m}) q_i^{\tilde{m}}(z_i) \frac{A^{\tilde{m},m}}{A_i^{\tilde{m},m}} \right]^{-1}. \tag{31}$$

The dependence of $\mathcal{L}_{\mathcal{Z}}$ on $q^0(m)$ is particularly simple: $\mathcal{L}_{\mathcal{Z}}(q, w, \nu) \approx \sum_m q^0(m) \mathcal{E}(m) - T\big(S(q^0)+1\big)$ up to q^0-independent terms and where

$$\mathcal{E}(m) \triangleq \mathbb{E}_{q^m}(G) - T \left(S[q^m] + B^{m,m} - \sum_{\tilde{m}} A^{m,\tilde{m}} \nu_{\tilde{m}} + \ln \nu_m \right),$$

Thus, the mixture weights are Boltzmann distributed with energy function $\mathcal{E}(m)$:

$$q^0(m) = \frac{\exp\bigl(-\mathcal{E}(m)/T\bigr)}{\sum_{\tilde{m}} \exp\bigl(-\mathcal{E}(\tilde{m})/T\bigr)}. \tag{32}$$

The determination of $q_i^m(z_i)$ is similar. The relevant terms in $\mathcal{L}_{\mathcal{Z}}$ involving $q_i^m(z_i)$ are $\mathcal{L}_{\mathcal{Z}} \approx q^0(m) \sum_{z_i} \mathcal{E}_m(z_i) q_i^m(z_i) - TS(q_i^m)$ where

$$\mathcal{E}_m(z_i) = \mathbb{E}_{q_{-i}^m}(G|z_i) - T\left(\ln w_i(z_i|m) - \sum_{\tilde{m}} \frac{A^{m,\tilde{m}}}{A_i^{m,\tilde{m}}} \nu_{\tilde{m}} w_i(z_i|\tilde{m})\right).$$

As before the conditional expectation $\mathbb{E}_{q_{-i}^m}(G|z_i)$ is $\sum_{z_{-i}} G(z_i, \mathbf{z}_{-i}) q_{-i}^m(\mathbf{z}_{-i})$. The mixture probabilities are thus determined as

$$q_i^m(z_i) = \frac{\exp\bigl(-\mathcal{E}_m(z_i)/T\bigr)}{\sum_{z_i} \exp\bigl(-\mathcal{E}_m(z_i)/T\bigr)}. \tag{33}$$

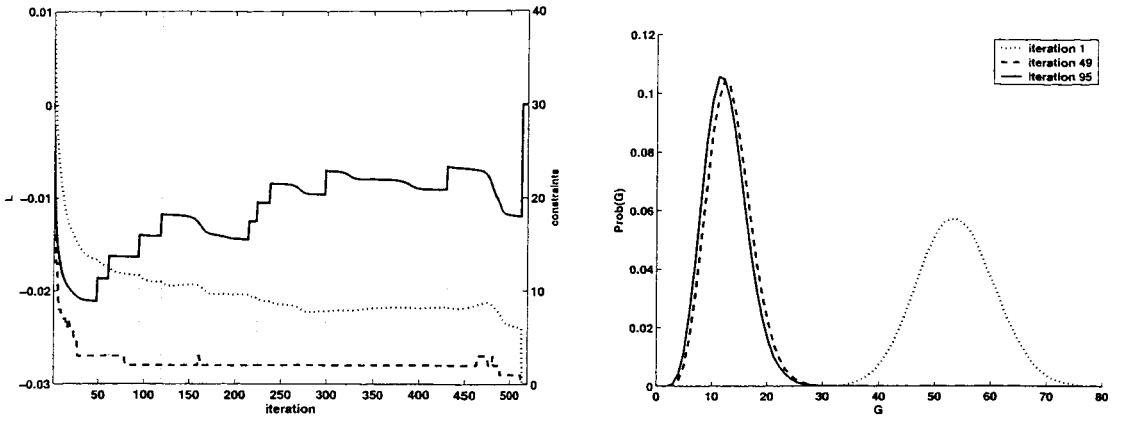
6.1 Agent Communication

These results require minimal communication between agents. An agent, call this the 0-agent, is assigned to manage the determination of $q^0(m)$, and (i,m)-agents manage the determination of $q_i^m(z_i)$. The M (i,m)-agents for a fixed i communicate their $w_i(z_i|m)$ to determine $A_i^{m,\tilde{m}}$. These results along with the $B_i^{m,m}$ from each (i,m) agent are then forwarded to the 0-agent who forms $A^{m,\tilde{m}}$ and $B^{m,m}$ broadcasts this back to all (i,m)-agents. With these quantities and the local estimates for $\mathbb{E}_{q_{-i}^m}(G|z_i)$, all q_i^m can be updated independently.

7 Experiments

In this section we demonstrate our methods on some simple problems. To keep this already lengthy paper from being too large, this section is meant to be illustrative only. The reader is directed to (Bieniawski & Wolpert 2004*a*, Bieniawski et al. 2004, Bieniawski & Wolpert 2004*b*, Lee & Wolpert 2004, Wolpert & Lee 2004, Macready & Wolpert 2004*a*) for many other related experiments.

As our first example we test the probability collective method on two different problems: a k-sat constraint satisfaction problem having multiple feasible solutions, and optimization of an unconstrained optimization of an NK function.



(a) (b)

Figure 3: (a) Evolution of Lagrangian value (solid line), expected constraint violation (dotted line), and constraint violations of most likely configuration (dashed line). (b) $P(G)$ after minimizing the Lagrangian for the first 3 multiplier settings. At termination $P(G) = \delta(G)$.

7.1 k-sat

The k-sat problem is perhaps the best studied CSP (Mezard, Parisi & Zecchina 2002). The goal is to assign N binary variables z_i values so that C clauses are satisfied. The ath clause involves k variables labelled by $v_{a,j} \in [1, N]$ (for $j \in [1, k]$), and k binary values associated with each a and labelled by $\sigma_{a,j}$. The ath clause is satisfied iff $\bigvee_{j=1}^{k} [z_{v_{a,j}} = \sigma_{a,j}]$ is true so we define the ath constraint as

$$c_a(\mathbf{z}) = \begin{cases} 0 & \text{if } \bigvee_{j=1}^{k} [z_{v_{a,j}} = \sigma_{a,j}] \\ 1 & \text{otherwise} \end{cases}.$$

As the ath clause is violated only when all $z_{v_{a,j}} = \overline{\sigma}_{a,j}$ (with $\overline{\sigma} \triangleq \text{not}\, \sigma$), the Lagrangian over product distributions can be written as $\mathcal{L}(q) = \boldsymbol{\lambda}^{\top} \mathbf{c}(q) - TS(q)$ where $\mathbf{c}(q)$ is the C-vector of expected constraint violations whose ath component is $c_a(q) \triangleq \sum_{\mathbf{z}} c_a(\mathbf{z}) q(\mathbf{z}) = \prod_{j=1}^{k} q_{v_{a,j}}(\overline{\sigma}_{a,j})$, and $\boldsymbol{\lambda}$ is the C vector of Lagrange multipliers. The only communication required to evaluate G and its conditional expectations is between agents appearing in the same clause. Typically, this communication network is sparse; for the $N = 100$, $k = 3$, $C = 430$ variable problem we consider each agent interacts with only 6 other agents on average.

We first present results for a single product distribution. For any fixed setting of the Lagrange multipliers, the Lagrangian is minimized by iterating Eq. (9). Had the minimization been done by

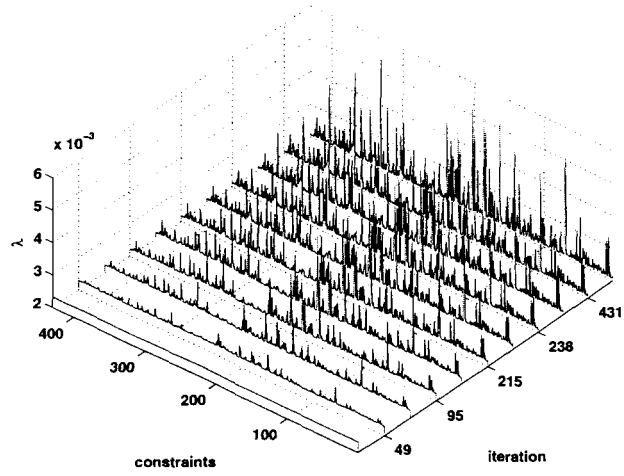


Figure 4: Each constraint's Lagrange multiplier versus the iterations when they change.

the Brouwer method, any random subset of variables, no two of which appear in the same clause, could be updated simultaneously while still ensuring that the Lagrangian would decrease at each iteration.

The minimization is terminated at a local minimum q^*. If all constraints are satisfied at q^* we return the solution $\mathbf{z}^* = \arg\max_{\mathbf{z}} q^*(\mathbf{z})$ otherwise the Lagrange multipliers are updated according to Eq. (10). In the present context, this updating rule offers a number of benefits. Firstly, those constraints which are violated most strongly have their penalty increased the most, and consequently, the agents involved in those constraints are most likely to alter their state. Secondly, the Lagrange multipliers contain a history of the constraint violations (since we keep adding to $\boldsymbol{\lambda}$) so that when the agents coordinate on their next move they are unlikely to return a previously violated state. This mimics the approach used in taboo search where revisiting of configurations is explicitly prevented, and aids in an efficient exploration of the search space. Lastly, rescaling the Lagrangian after each update of the multipliers by $\mathbf{1}^\top\boldsymbol{\lambda} = \sum_a \lambda_a$ gives $\mathcal{L}(q) = \hat{\boldsymbol{\lambda}}^\top \mathbf{c}(q) - \hat{T}S(q)$ where $\hat{\boldsymbol{\lambda}} = \boldsymbol{\lambda}/\mathbf{1}^\top\boldsymbol{\lambda}$ and $\hat{T} = T/\mathbf{1}^\top\boldsymbol{\lambda}$. Since $\sum_a \hat{\lambda}_a = 1$ the first term reweights clauses according to their expected violation, while the temperature $\hat{T}$ cools in an automated way as the Lagrange multipliers increase. Cooling is most rapid when the expected constraint violation is large and slows as the optimum is approached. The parameters α_λ^t thus govern the overall rate of cooling. We used the fixed value $\alpha_\lambda^t = 0.5$.

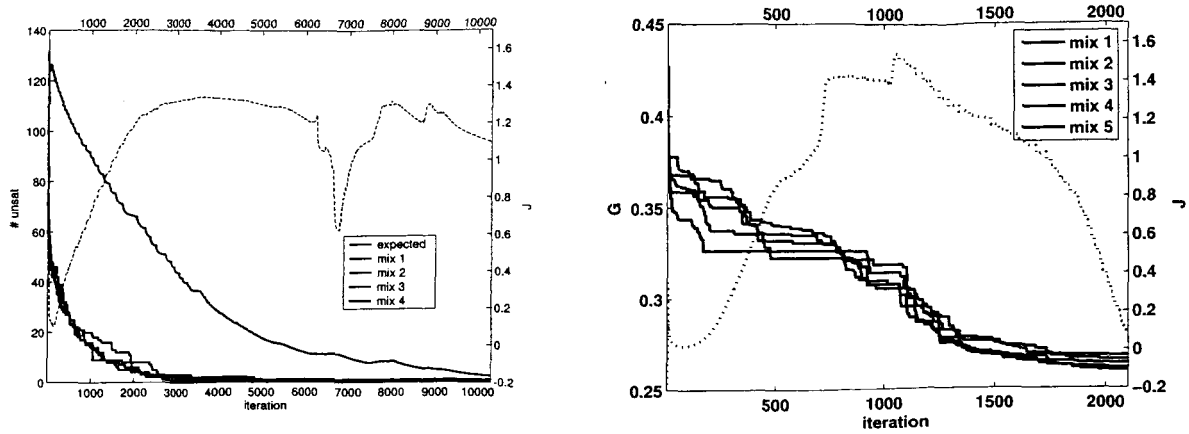
Figure 3 presents results for a 100 variable $k = 3$ problem using a single mixture. The problem

is satisfiable formula `uf100-01.cnf` from SATLIB (`www.satlib.org`). It was generated with the ratio of clauses to variables being near the phase transition, and consequently has few solutions. Fig. 3(a) shows the variation of the Lagrangian, the expected number of constraint violations, and the number of constraints violated in the most probable state $z_{\rm mp} \triangleq \arg\max_{\mathbf{z}} q(\mathbf{z})$ as a function of the number of iterations. The starting state is the maximum entropy configuration, and the starting temperature is $T = 1.5 \cdot 10^{-3}$. The iterations at which the Lagrange multipliers are updated are indicated by vertical dashed lines which are clearly visible as discontinuities in the Lagrangian values. To show the stochastic underpinnings of the algorithm we plot in Fig. 3(b) the probability density of the number of constraint violations obtained as $P(G) = \sum_{\mathbf{z}} q(\mathbf{z})\delta\big(G - \sum_a c_a(\mathbf{z})\big)$.[17] Figure 4 shows the evolution of the renormalized Langrange multipliers $\hat{\boldsymbol{\lambda}}$. At the first iteration the multiplier for all clauses are equal. As the algorithm progresses weight is shifted amongst difficult to satisfy clauses.

Results on a larger problem with multiple mixtures are shown in Fig. 5(a). This is the 250 variable/1065 clause problem `uf250-01.cnf` from SATLIB with the first 50 clauses removed so that the problem has multiple solutions. The optimization was performed by selecting a random subset of variables, no two of which appear in the same clause at each iteration, and updating according to Eqs. (30), (31), (32), and (33). After convergence the Lagrange multipliers are updated. The initial temperature is 10^{-1}. We plot the number of constraints violated in the most probable state of each mixture as a function of the number of updates. as well as the expected number of violated constraints. After 8000 steps three distinct solutions have been found along with a fourth configuration which violates a single constraint.

7.2 Minimization of NK Functions

The NK model defines a family of tunably difficult optimization problems (Kauffman & Levin 1987). The objective of N binary variables is defined as the average of N contributions local to each variable z_i and involving $0 \geq K \geq N-1$ other randomly chosen variables $z_i^1 \cdots z_i^K$: $G(\mathbf{z}) = N^{-1}\sum_{i=1}^{N} E_i(z_i; z_i^1, \cdots z_i^K)$. For each of the 2^{K+1} local configurations E_i is assigned a value drawn uniformly from 0 to 1. K controls the number of local minima; under Hamming neighborhoods $K = 0$ optimization landscapes have a single global optimum and $K = N-1$ landscapes have on average $2^N/(N+1)$ local minima. Further properties of NK landscapes may



(a) (b)

Figure 5: (a) The solid curves show the number of unsatisfied clauses in the most probable configuration $z_{\rm mp}$ of each of the 4 mixtures vs iterations. The topmost solid black line plots the expected number of violations, and the dashed black line shows the approximation to the JS distance. (b) The solid curves show the evolution of the G value of the best $z_{\rm mp}$ configurations for each of 5 mixtures versus number of iterations. The dashed black line shows the corresponding approximation to the JS distance.

be found in (Durrett & Limic 2001). Fig. 5(b) plots the energy of a 5 mixture model for a multimodal $N = 300$ $K = 2$ function. The $K-1$ spins other than i upon which E_i depends were selected at random. At termination of the PC algorithm (at a small but non-zero temperature), five distinct configurations are obtained with the nearest pair of solutions having Hamming distance 12. Note that unlike the k sat problem which has multiple configurations all having the same global minimal energy, the JS distance (the dashed curve) of Fig. 5(b) drops to zero as the temperature decreases. This is because at exactly zero temperature there is no term forcing different solutions, and the Lagrangian is minimized by having all mixtures converge to delta functions at the lowest objective configuration.

8 Relation of PC to other work

There has been much work from many fields related to PC. The maxent Lagrangian has been used in statistical physics for over a century under the rubric of "free energy". Its derivation in terms of information theoretic statistical inference was by Jaynes (Jaynes 1957). The maxent Lagrangian has also appeared occasionally in game theory as a heuristic, without a statistical inference justification

(be it information-theoretic or otherwise) (Fudenberg & Levine 1998, Shamma & Arslan 2004).[18] In none of this earlier work is there an appreciation for its relationship with the related work in other fields.

In the context of distributed control/optimization, the distinguishing feature of PC is that it does not view the variable $\mathbf{x}$ as the fundamental object, but rather the distribution across it, q. Samples of that distribution are not the direct object of interest, and in fact are only used if necessary to estimate functionals of q. The fundamental objective function is stated in terms of q. As explicated in the next subsection, the associated optimization algorithms are related to some work in several fields. Heretofore those fields have been unaware of each other, and of the breadth of their relation to information theory and game theory.

Finally, we note that the maxent or qp Lagrangian $\mathcal{L}(q) = E_q(G) - TS(q)$ can be viewed as a barrier-function (interior point) method with objective $E_q(G)$. An entropic barrier function is used to enforce the constraints $q_i(x_i) \geq 0 \; \forall i$ and x_i, with the constraint that all q_i sum to 1 being implicit.

8.1 Various schemes for updating q

We have seen that the qp Lagrangian is minimized by the product distribution q given by

$$q_i(x_i) \propto \exp\bigl(-E_{q_{-i}}(G|x_i)/T\bigr). \tag{34}$$

Direct application of these equations that minimize the Lagrangian form the basis of the Brouwer update rules. Alternatively, steepest descent of the maxent Lagrangian forms the basis of the Nearest Newton algorithm. These update rules have analogues in conventional (non-PC) optimization. For example, Nearest-Newton is based on Newton's method, and Brouwer updating is similar to block-relaxation. This is one of the advantages of embedding the original optimization problem involving x in a problem involving distributions across x: it allows us to solve problems over non-Euclidean (e.g., countable) spaces using the powerful methods already well-understood for optimization over Euclidean spaces.

However there are other PC update rules that have no direct analogue in such well-understood methods for Euclidean space optimization. Algorithms may be developed that minimize the pq

Lagrangian $D(p_T \| q)$ where $p_T(\mathbf{x}) = \exp\bigl(-G(\mathbf{x})/T\bigr)/Z(T)$ with $Z(T)$ being the normalization of the Boltzmann distribution. The *pq* Lagrangian is minimized by the the product of the marginals of the Boltzmann distribution, i.e. $q_i(x_i) = \int d\mathbf{x}_{-i}\, p_T(\mathbf{x})$. Another example of update rules without Euclidean analogues are the iterative focusing update rules described in (Wolpert 2004*b*). Iterative focusing updates are intrinsically tied into the fact that we're minimizing (the distribution setting) an expectation value.

A subset of update rules arising from *qp* and *pq* Lagrangians are described in (Wolpert 2004*b*). In all cases, the updates may be written as multiplicative updating of q. The following is a list of the update ratios $r_{q,i}(x_i) \equiv q_i^{t+1}(x_i)/q_i^t(x_i)$ of some of those rules. In all of these, F_G is a probability distribution over x that never increases between two x's if G does (e.g., a Boltzmann distribution in $G(x)$). In addition, const is a scalar that ensures the new distribution is properly normalized and α is a stepsize.[19]

Gradient descent of qp distance to F_G:

$$1 - \alpha\Biggl\{\frac{\mathbb{E}_{q^t}(\ln F_G | x_i) + \ln(q_i^t(x_i))}{q_i^t(x_i\ )}\Biggr\} - \frac{\text{const}}{q_i^t(x_i)} \tag{35}$$

Nearest Newton descent of qp distance to F_G:

$$1 - \alpha\bigl\{\mathbb{E}_{q^t}(\ln F_G | x_i) + \ln(q_i^t(x_i))\bigr\} - \text{const} \tag{36}$$

Brouwer updating for qp distance to F_G:

$$\text{const} \times \frac{e^{\mathbb{E}_{q^t}(\ln F_G | x_i)}}{q_i^t(x_i)} \tag{37}$$

Importance sampling minimization of pq distance to F_G:

$$\text{const} \times \mathbb{E}_{q^t}\left(\frac{F_G}{q^t} | x_i\right) \tag{38}$$

Iterative focusing of $\tilde{q}$ with focusing function F_G using qp distance and gradient descent:

$$1 - \alpha \left\{ \frac{\mathbb{E}_{q^t}(\ln F_G | x_i) + \ln \frac{q_i^t(x_i)}{\tilde{q}_i(x_i)}}{q^t(x_i)} \right\} - \frac{\text{const}}{q^t(x_i)} \tag{39}$$

Iterative focusing of $\tilde{q}$ with focusing function F_G using qp distance and Nearest Newton:

$$1 - \alpha \left\{ \mathbb{E}_{q^t}(\ln F_G | x_i) + \ln \frac{q_i^t(x_i)}{\tilde{q}_i(x_i)} \right\} - \text{const} \tag{40}$$

Iterative focusing of $\tilde{q}$ with focusing function F_G using qp distance and Brouwer updating:

$$\text{const} \times e^{\mathbb{E}_{q^t}(\ln F_G | x_i)} \times \frac{\tilde{q}(x_i)}{q_i^t(x_i)} \tag{41}$$

Iterative focusing of $\tilde{q}$ with focusing function F_G using pq distance:

$$\text{const} \times \mathbb{E}_{\tilde{q}}(F_G | x_i) \times \frac{\tilde{q}(x_i)}{q_i^t(x_i)} \tag{42}$$

Note that some of these update ratios are themselves proper probability distributions, e.g., the Nearest Newton update ratio.

This list highlights the ability to go beyond conventional Euclidean optimization update rules, and is an advantage of embedding the original optimization problem in a problem over a space of probability distributions. Another advantage is the fact that the distribution itself provides much useful information (e.g., sensitivity information). Yet another advantage is the natural use of Monte Carlo techniques that arise with the embedding, and allow the optimization to be used for adaptive control.

There are also overlaps of PC with evolutionary approaches to optimization (e.g., genetic algorithms). Many techniques in the evolutionary computation community use Boltzmann distributions in ad hoc ways to update a "population" of $\mathbf{x}$'s, e.g., "truncation selection" and "Boltzmann selection" (G.Ficici, Melnik, & Pollack 2000). These rules are not formally derived, and do not directly concern themselves with distributions q. However some of them are similar to the PC update rules, in particular iterative focusing rules, only applied to sets of Monte Carlo sample points (the

population) rather than to q. There are other techniques which also use Boltzmann distributions and samples, although without a multi-member population, e.g., simulated annealing.

These early techniques do not consider the underlying distribution that gets sampled to produce the population. Such consideration was introduced in PBIL (Baluja 2002), MIMIC (Bonet, Jr. & Viola 1996) and other EDA algorithms (Larraaga & Lozano 2001), followed shortly by the powerful CE method (Rubinstein & Kroese 2004).

However while considering distributions, none of this early work casts the objective as a minimization of a functional of that distribution. Accordingly, all the power arising from minimizing Euclidean vectors is absent in this work. There is none of the second order methods, difference utilities, or data-ageing that appear to be crucial for very large problems. Nor does this earlier work exploit semicoordinate transformations (the topic of this paper), oracle-based methods (Lee & Wolpert 2004), etc. Despite this, the results in (Rubinstein & Kroese 2004) in particular are compelling. (Note that the CE method, as applied, is identical to Eq. (42), although with a different justification.)

There is other previous work on optimization that has directly considered the distribution q as the object of interest. In particular deterministic annealing (Duda et al. 2000) is "bare-bones" parallel Brouwer updating. This involves no data-aging (or any other scheme to avoid thrashing of the agents), difference utilities, etc..[20]

Most tantalizingly, probability matching (Sabes & Jordan 1996) uses Monte Carlo sampling to optimize a functional of q. However this work was in the context of a single agent, did not exploit techniques like data-ageing, and was not pursued.

Other work has both viewed q as the fundamental object of interest and used techniques like data-aging and difference utilities (Wolpert, Tumer & Frank 1999, Wolpert et al. 2003, Wolpert 2003*c*). However this work was not based on information-theoretic considerations and had no explicit objective function for q. It was the introduction of such considerations that resulted in PC.

Finally, shortly after the introduction of PC a variant of its Monte Carlo version of parallel Brouwer updating has been introduced, called the MCE method (Rubenstein 2005). In this variant the annealing of the Lagrangian doesn't involve changing the temperature β, but instead changing the value of a constraint specifying $\mathbb{E}_q(G)$. Accordingly, rather than jump directly to the (β-specified) solution given above, one has to solve a set of coupled nonlinear equations relating all

the q_i. (Another distinguishing feature is no data-ageing, difference utilities or the like are used in the MCE method.) The MCE method has been justified with the KL argument reviewed above rather than with "ratchet"-based maximum entropy arguments. This has redrawn attention to the role of the argument-ordering of the KL distance, and how it relates Brouwer updating and the CE method.

9 Conclusion

A distributed constrained optimization framework based on probability collectives has been presented. Motivation for the framework was drawn from an extension of full-rationality game theory to bounded rational agents. An algorithm that is capable of obtaining one or more solutions simultaneously was developed and demonstrated on two problems. The results show a promising, highly distributed, off-the-shelf approach to constrained optimization.

There are many avenues for future exploration. Alternatives to the Lagrange multiplier method used here can be developed for constraint satisfaction problems. By viewing the constraints as separate objectives, a Pareto-like optimization procedure may be developed whereby a gradient direction is chosen which is constrained so that no constraints are worsened. This idea is motivated by the highly successful WalkSAT (Selman, Kautz & Cohen 1996) algorithm for k-sat in which spins are flipped only if no previously satisfied clause becomes unsatisfied as a result of the change.

Probability collectives also offer promise in devising new methods for escaping local minima. Unlike traditional optimization methods where monotonic transformations of the objective leave local minima unchanged, such transformations will alter the local minima structure of the Lagrangian. This observation, and alternative Lagrangians (see (Rubinstein 2001) for a related approach using a different minimization criterion) offer new approaches for improved optimization.

ACKNOWLEDGEMENTS We would like to thank Charlie Strauss for illuminating conversation and Bill Dunbar and George Judge for helpful comments on the manuscript.

Notes

[1] $|\mathcal{S}|$ denotes the number of elements in the set $\mathcal{S}$.

[2] In this paper vectors are indicated in bold font and scalars are in regular font.